DESARROLLO DE UNA HERRAMIENTA PARA LA EDICIÓN DIGITAL DE FICCIÓN INTERACTIVA MEDIANTE EL PARADIGMA DE TEXTO EXTENSIBLE

Mª COVADONGA DÍEZ SANMARTÍN

MÁSTER EN INGENIERÍA INFORMÁTICA, FACULTAD DE INFORMÁTICA,
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin Máster en Sistemas Inteligentes
Febrero 2018

Directores:
José Luis Sierra Rodríguez
Antonio Sarasa Cabezuelo

Calificación: 9 – Sobresaliente

Autorización de Difusión

Mª COVADONGA DÍEZ SANMARTÍN

Fecha: 21/02/2018

La abajo firmante, matriculada en el Máster en Ingeniería Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autora el presente Trabajo Fin de Máster: "DESARROLLO DE UNA HERRAMIENTA PARA LA EDICIÓN DIGITAL DE FICCIÓN INTERACTIVA MEDIANTE EL PARADIGMA DE TEXTO EXTENSIBLE", realizado durante el curso académico 2016-2017 bajo la dirección de José Luis Sierra Rodríguez y Antonio Sarasa Cabezuelo en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en internet y garantizar su preservación y acceso a largo plazo.

Resumen

En la última década, el libro digital se ha convertido en una alternativa real al libro físico. Desde el punto de vista técnico, se han desarrollado diferentes especificaciones para crear libros digitales, así como herramientas de edición. Las posibilidades de interacción que ofrece un libro digital en comparación con un libro físico son importantes, dado que el escritor tiene potencialmente, todas las posibilidades que ofrecen las nuevas tecnologías. Sin embargo, la realidad es que la mayoría de los editores de libros digitales permiten implementar un número limitado de funciones en un libro digital, tales como buscar en un diccionario o realizar una anotación. Esta situación se debe en parte a las posibilidades de reproducción de los dispositivos de lectura (eReaders) que solo son capaces de ejecutar y procesar las funciones antes mencionadas, a las especificaciones de libros digitales que no contemplan otro tipo de acciones sobre un libro digital y a las propias herramientas de edición digital que están diseñadas para editar los libros digitales con las posibilidades comentadas anteriormente. Es por ello, que supone un gran reto el ampliar las posibilidades de interacción que se pueden implementar sobre un libro digital y de esta manera enriquecer la experiencia lectora.

Este trabajo de investigación, se plantea en el contexto del reto comentado. Así, el objetivo que se quería cubrir en el trabajo consistía en investigar cómo ampliar las funcionalidades que tiene a su disposición un editor de libros digitales en las herramientas de edición, para enriquecer la interacción y la experiencia lectora. Con este fin, se ha realizado una labor de investigación para analizar qué tipo de necesidades de interacción podrían ser interesantes para un lector, qué tipo de tecnologías serían necesarias para su implementación y cómo se podrían adaptar para que fueran reproducibles con la tecnología actual de eReaders. Como resultado de este trabajo, se ha creado una herramienta de edición de libros digitales de tipo WYSIWYG denominada ILSAditor que facilita la creación de libros digitales en los que se pueden incluir algunos elementos orientados a enriquecer la experiencia del lector basados en las propuestas definidas en el ámbito de la *ficción interactiva*. La herramienta está diseñada de una forma muy intuitiva. Así, el escritor puede crear de una manera simple elementos de ficción interactiva, exportar texto interactivo a un formato orientado a la web (HTML5, JavaScript y CSS) u orientado a

dispositivos electrónicos (EPUB). Los elementos de ficción interactiva soportados se circunscriben al uso de *texto extensible* (*stretchtext*), permitiendo al editor de un libro digital organizar el contenido del mismo en varias capas de profundización o itinerarios que pueden ser gestionados a través de la ocultación o develación del texto. Esto permite que el texto se adapte a los diferentes niveles de lectura o profundización. De esta forma, se podría crear, por ejemplo, un mismo cuento dirigido a diferentes edades o niveles de comprensión en un único libro digital, gestionando cada nivel del mismo mediante texto extensible.

ILSAditor se ha desarrollado como una aplicación de escritorio basada en extender la funcionalidad del editor JavaScript TinyMCE, utilizando tecnología Java FX. El editor TinyMCE ha sido ampliamente utilizado en gestores de contenidos educativos como Moodle, sistemas de gestión de contenidos tan conocidos como Joomla o WordPress para soportar el paradigma de texto extensible de manera intuitiva y sencilla para los autores, y en aplicaciones como Evernote.

La memoria de este trabajo fin de máster describe la versión actual de la herramienta desarrollada, el proceso de diseño y desarrollo seguido, y la evaluación preliminar de su usabilidad en una experiencia piloto.

Palabras clave

Stretchtext, ficción interactiva, TinyMCE, Java FX, HMTL5, JavaScript, CSS, EPUB.

Abstract

In the last decade, digital book has become a real alternative to the physical book. From the technical point of view, different specifications have been developed to create digital books, as well as editing tools. The interaction possibilities offered by a digital book compared to a physical book are important, given that the writer potentially has all the possibilities offered by new technologies. However, the reality is that most digital book publishers allow to implement a limited number of functions in a digital book such as searching a dictionary or add notes. This situation is due in part to the reading possibilities of the e-book reader (eReaders) that are only capable of executing and processing the aforementioned functions, to the specifications of digital books that do not contemplate other types of actions on a digital book and the own digital edition tools that are designed to publish the digital books with the possibilities before commented. That is why it is a great challenge to expand the interaction possibilities that can be implemented on a digital book and, in this way, enrich the reading experience.

This research work is posed in the context of the aforementioned challenge. Thus, the objective to be covered in the work was to investigate how to extend the functionalities available to an editor of digital books in the editing tools to enrich the interaction and the reading experience. For this, a research work has been carried out to analyse what type of interaction needs could be interesting for a reader, what kind of technologies would be necessary for its implementation and how they could be adapted to be reproducible with the current eReaders technology. As a result of this work, a WYSIWYG digital book editing tool named ILSAditor has been created that helps the creation of digital books that can include some elements aimed at enriching the reader's experience based on the proposals defined in the field of interactive fiction. The tool is designed in a very intuitive way. Thus, the writer can create in a simple way interactive fiction elements, export interactive text to a web-oriented format (HTML5, JavaScript and CSS) or to electronic devices (EPUB). Supported interactive fiction elements are limited to the use of *stretchtext*, allowing the editor of a digital book to organize the book content into several *deepening layers* or *itineraries* that can be managed through the hiding or unveiling of the text. This allows the text to adapt to different levels of

reading or deepening. For example, one story could be created for different ages or comprehension levels in a single digital book, managing each level through *stretchtext*.

ILSAditor has been developed as a desktop application based on extending the functionality of the JavaScript editor TinyMCE, which, uses Java FX technology. The TinyMCE editor has been widely used in learning management systems such as Moodle or in content management systems as Joomla or WordPress, to support the *stretchtext* paradigm in an intuitive and simple way for the authors and applications as Evernote.

This degree project for Master of Science Engineering describes the current version of the developed tool, the design and development process followed, and the preliminary evaluation of its usability in a pilot experience.

Keywords

Stretchtext, interactive fiction, TinyMCE, Java FX, HMTL5, JavaScript, CSS and EPUB.

Índice de contenidos

Agradecimientos

Agradezco a los profesores José Luis Sierra y a Antonio Sarasa, miembros del grupo de investigación ILSA de la Facultad de Informática de la UCM, por su colaboración y su dedicación en la dirección de mi Trabajo fin de Máster.

Agradezco también a la profesora María Goicoechea, coordinadora del proyecto Elite y responsable del grupo de investigación LEETHI de la facultad de Filología de la UCM, por sus valiosos comentarios y aportaciones durante el desarrollo de la herramienta ILSAditor.

Me gustaría también agradecer a Joaquín Gayoso, miembro del grupo de investigación ILSA, por su ayuda y valiosos consejos.

Expreso también mi gratitud a Laura Sánchez (grupo de investigación LEETHI), Amelia del Rosario (Departamento de Filología Francesa, Facultad de Filología de la UCM), Mª Dolores Romero (responsable del grupo LOEP de la Facultad de Filología de la UCM) y a Elena Bárcena (responsable del grupo ATLAS de la Facultad de Filología de la UNED) por su colaboración en el desarrollo de la herramienta ILSAditor.

Deseo expresar un especial agradecimiento a mi madre por su incondicional apoyo, a mi querido Andrés por su inestimable ayuda, a mi abuela por ser mi fuente de inspiración y a mi familia y amigos por sus ánimos.

Finalmente, agradezco a la Comunidad de Madrid que, a través de la Dirección General de Universidades e Investigación de la Consejería de Educación, Juventud y Deporte me ha permitido disfrutar de una beca de colaboración financiada íntegramente a través del proyecto Elite (Edición literaria electrónica), número de referencia del proyecto: H2015/HUM-3426.

Chapter 1 - Introduction

Digital books have become a real alternative to books in physical format. Currently, a digital book implements an electronic text version of the book supplemented with a set of additional services such as the ability to search for words in an online dictionary or the inclusion of digital annotations in parts of its content. However, the current digital books do not take advantage of all the possibilities offered by the digital world, and if they were used, would result in a richer experience for the reader. Among all the initiatives that aim to improve the reader's experience of a digital book, interactive fiction stands out. It is a set of techniques that extend the classic narrative media to directly involve spectators during the story evolution [Donikian et al, 2004]. Interactive fiction techniques include *stretchtext*. As defined in [Brusilovsky P., 1997], *stretchtext* is a special type of hypertext in which the result of clicking on a word or phrase on the hyperlink allow to extend the text on the same page (contrary to the usual behavior, which usually causes the context switch to a new page or region of the text). Thus, *stretchtext* allows the writer of a book or another type of digital work, organize it in several levels or physical layers that can be managed by the actions of contracting or expanding the text. In this way, as discussed in [Boyle C and Encarnacion A, 1994], the advantage of this behavior, based on the concealment or unveiling of the text, is that the reader no longer needs to adapt to the document but is the document that adapts to the reader, allowing information to be presented at different levels of deepening or hierarchies. So, for example, you can create a story aimed at different levels of understanding or ages in a single book, managing each version of it by using *stretchtext*.

This work has focused on the study of how to expand the functionalities offered by a digital book with the aim of enriching its reading using interactive fiction, and its implementation in an experimental software tool.

In the following sections, the goals of the work are presented, as well as the work plan that has been followed.

1.1 Goals

The goals set out in this work have been:

- Analyze from a technological point of view, the different initiatives and techniques aimed at improving the reading experience in digital books, focusing on interactive fiction.
- Design and develop a digital book editing tool that implements some of the interactive fiction techniques studied.
- Evaluate the tool in a pilot experience with the objective of knowing the usability and acceptance by readers and writers of digital books.

1.2 Working Plan

To carry out this project the following planning has been followed:

- The first phase of the work consisted of studying the most important initiatives in the digital literature and in the processes of digital publication as well as the software technologies involved. In particular, its possible implementation using HTML5 has been analysed. The results of this phase have been described in chapter two of this work.
- The second phase of the work has focused on the design and development of a tool prototype that implements some of the interactive fiction techniques analysed. In chapter three the developed tool is presented, its architecture and technologies used are described, as well as its construction has been refined.
- In the last stage of this work development, a preliminary evaluation of the tool usability has been carried out by a pilot use experience. Chapter four describes the experience and analyses the results obtained.

Finally, chapter five discusses the results obtained from the realization of this project and present the main conclusions and future research lines of derived from it.

Capítulo 1 - Introducción

Los libros digitales se han convertido en una alternativa real a los libros en formato físico. Actualmente, un libro digital implementa una versión en texto electrónico del libro complementado con un conjunto de servicios adicionales tales como la posibilidad de buscar palabras en un diccionario en línea o la inclusión de anotaciones digitales en partes de su contenido. Sin embargo, los actuales libros digitales no aprovechan todas las posibilidades que ofrece el mundo digital, y si fueran usadas, redundarían en una experiencia más rica para el lector. De entre todas las iniciativas que tienen como objetivo mejorar la experiencia del lector de un libro digital, destaca la ficción interactiva. Se trata de un conjunto de técnicas que extienden a los medios narrativos clásicos para involucrar directamente a los espectadores durante la evolución de la historia [Donikian et al, 2004]. Entre las técnicas de ficción interactiva se encuentra el *texto extensible* o *stretchtext*. [Brusilovsky P., 1997] define el *stretchtext como* un tipo especial de hipertexto en el que el resultado de pulsar sobre un hiperenlace asociado a una palabra o frase permite extender el texto en la misma página (al contrario del comportamiento más habitual, que provoca el cambio de contexto a una nueva página o región del texto). De esta forma, el *stretchtext* permite al escritor de un libro digital, organizar el contenido textual del libro en varias capas físicas que pueden ser gestionadas mediante las acciones de contraer o expandir el texto. Tal como se analiza en [Boyle C y Encarnacion A, 1994], la ventaja de esta técnica, basada en ocultar o develar el texto a voluntad del lector, es que éste ya no necesita adaptarse al documento, sino que es el documento el que se adapta al lector, permitiendo que la información se presente en distintos niveles de profundización o jerarquías. Así, por ejemplo, se puede crear un cuento dirigido a diferentes niveles de comprensión o edades en un único libro, gestionando cada versión del mismo mediante *stretchtext*.

Este trabajo, se ha centrado en el estudio de cómo ampliar las funcionalidades que ofrece un libro digital con el objetivo de enriquecer su lectura usando ficción interactiva, y su implementación en una herramienta software experimental.

En las siguientes secciones se plantean los objetivos del trabajo, así como el plan de trabajo que se ha seguido.

1.1. Objetivos

Los objetivos planteados en este trabajo han sido:

- Analizar desde un punto de vista tecnológico, las diferentes iniciativas y técnicas orientadas a mejorar la experiencia lectora en los libros digitales, centrándose en la ficción interactiva.
- Diseñar y desarrollar una herramienta de edición de libros digitales que implemente algunas de las técnicas de ficción interactiva estudiadas.
- Evaluar la herramienta en una experiencia piloto con el objetivo de conocer la usabilidad y aceptación por parte de los lectores y los escritores de libros digitales.

1.2. Plan de Trabajo

Para llevar a cabo este proyecto se ha seguido el siguiente plan de trabajo:

- La primera fase del trabajo ha consistido en estudiar las iniciativas más importantes en la literatura digital y en los procesos de publicación digital, así como las tecnologías software involucradas. En este sentido, una parte importante de la revisión realizada se ha centrado en estudiar las técnicas de ficción interactiva y en las tecnologías software necesarias para su implementación. En particular, se ha analizado su posible implementación utilizando HTML5. Los resultados de esta fase se han descrito en el capítulo dos de la memoria.
- La segunda fase del trabajo se ha centrado en el diseño y desarrollo de un prototipo de herramienta que implementa algunas de las técnicas de ficción interactiva analizadas. En el capítulo tres se presenta la herramienta desarrollada, se describe su arquitectura y tecnologías utilizadas, y cómo se ha refinado su construcción.
- En la última etapa del desarrollo de este trabajo, se ha llevado a cabo una evaluación preliminar de la usabilidad de la herramienta mediante la realización de una

experiencia piloto de uso. El capítulo cuatro describe la experiencia realizada y analiza los resultados obtenidos.

El capítulo cinco discute los resultados obtenidos en este trabajo, y presenta las principales conclusiones y líneas futuras de trabajo derivadas del mismo.

Capítulo 2 - Estado de la Cuestión

En este capítulo se realiza una revisión del estado de la cuestión sobre literatura digital y sobre las herramientas tecnológicas que la soportan.

En este sentido, el capítulo se organiza en tres secciones. En la primera se dedica a revisar el dominio de la literatura digital focalizándolo en el tema de la ficción interactiva y sus extensiones. La segunda sección se dedica a revisar las principales herramientas y técnicas informáticas que se han utilizado para implementar la literatura digital. La última sección consiste en una discusión de las ideas descritas en las dos secciones anteriores.

2.1 Literatura Digital

Esta sección se dedica a revisar el concepto de literatura digital y las iniciativas más importantes que han surgido alrededor de la misma. Para ello, la exposición se organiza introduciendo los principales conceptos relativos a la literatura digital y a sus distintos géneros. Seguidamente se revisan con mayor detalle los conceptos relativos a la ficción interactiva. A continuación, se analiza el papel básico jugado por el hipertexto en literatura digital en general, y en ficción interactiva en particular. Por último, se realiza una introducción a la técnica de *stretchtext* como mecanismo para implementar la ficción interactiva.

2.1.1 Literatura Digital

La *literatura digital*, *ciberliteratura* o *literatura electrónica* es literatura explícitamente concebida para su consumo en un ordenador o en otro dispositivo electrónico de características similares. A este respecto, [Hayles, 2007] define las obras de literatura digital como objetos digitales creados mediante un soporte digital y generalmente destinados a leerse en un dispositivo electrónico. Esta definición excluye la literatura impresa digitalizada, debido a que el proceso de digitalización no es más que una conversión electrónica de obras literarias convencionales. Por el contrario, tal y como reconoce la ELO (*Electronic Literature Organization*) [ELO], las obras en literatura digital son obras con importantes aspectos literarios que aprovechan las capacidades y los contextos proporcionados por un solo ordenador o por una

red de ordenadores. La literatura digital se concibe, por tanto, para aprovechar las posibilidades de creación que ofrece la digitalización. Por lo tanto, sólo se pueden leer de forma adecuada en un soporte digital. De hecho, en [cervantesvirtual] se señala que los rasgos que diferencian la literatura digital de la literatura tradicional son el hipertexto, los recursos multimedia y la interactividad. Esta forma de literatura existe desde hace más de cincuenta años, comenzando con el poema de Théo Lutz's generado por ordenador titulado "Stochastische Texte" del año 1959. La historia de la literatura digital ha estado vinculada y enriquecida por la experimentación generada a partir de diversas formas de arte como las artes visuales, el arte sonoro, el cine o la literatura y está influenciada por los avances en las tecnologías electrónica e informática. Esta experimentación, es, por consiguiente, una forma híbrida de arte que requiere que sus lectores utilicen diferentes modalidades sensoriales como la vista, el tacto o el movimiento.

2.1.2 Géneros en Literatura Digital

La literatura digital se desarrolla notablemente en el mundo anglosajón desde la aparición de la obra descrita en [Joyce, 1987]. Esta obra se ofreció originalmente como una demostración del programa de creación y edición *Storyspace* [Storyspace]. Esta aplicación fue tan importante en la época para el desarrollo del campo de la literatura digital, que las obras creadas con esta herramienta, tales como las registradas en [Moulthrop, 1992] o [Jackson, 1995], se conocieron como las de la escuela de *Storyspace*. Estos primeros trabajos, llamados de *primera generación* u obras *clásicas* de la literatura digital [Hayles, 2007], se caracterizaban por contener bloques de texto con gráficos, animación, color y sonido limitados.

En los años siguientes con el desarrollo de Internet, surgieron nuevos programas de creación de obras literarias digitales y nuevos métodos de difusión: ya no era necesario distribuir las obras únicamente en CD, sino que estas podían distribuirse a través de la red, y aprovechar las características adicionales ofrecidas por los entornos de red (por ejemplo, colaboración entre múltiples usuarios). En estos nuevos escenarios, las limitaciones de *Storyspace* se hicieron patentes, como, por ejemplo, la imposibilidad de reproducir audio en la web o gráficos con

colores limitados por lo que fue paulatinamente perdiendo relevancia frente a las nuevas tecnologías.

En la segunda generación o período contemporáneo o postmoderno [Hayles, 2007], se utilizan variedades de esquemas de navegación y se usan las capacidades multimodales de la web. Surgen formas híbridas de literatura, como la obra de [Fisher, 2001] caracterizada porque los pasajes se ilustran con imágenes diferentes, la autora lee algunos en voz alta dando la oportunidad de escuchar el pasaje en lugar de leerlo y, además, incluye texto animado en su obra. Otros ejemplos son [Moulthrop, 1999], que contiene películas QuickTime y generación de texto aleatorio o [Memmott, 2000], programada con JavaScript y DHTML.

A principios de los 2000, surge una narrativa dependiente de las tecnologías móviles, desde la ficción corta distribuida en serie en teléfonos móviles o las narrativas dependientes de los medios basados en la localización. Así por ejemplo, en [Cardiff, 2000] la autora propone un recorrido espacial determinado por una narración, en [Cardiff, 2005] la obra incluye narraciones de audio y fotografía, en [Blast Theory, 2003] se consigue superponer una ciudad virtual con movimientos sobre espacios reales a través de la búsqueda de fotos escondidas en la realidad virtual, y en [Belanger y Petit, 2007] los participantes mapean de forma colectiva lugares en los que se sienten conectados emocionalmente.

Se produce también una apertura a la tercera dimensión, ya sea con interfaces gráficas de usuario apiladas añadiéndoles efectos de sombras, o con representaciones gráficas de espacios tridimensionales. La tercera dimensión puede ser una narración explorada a través de hipervínculos, enlaces lógicos de espacios de juegos en espacios tridimensionales o interfaces de realidad virtual. Algunos ejemplos son [Cayley, 2006], donde se presenta un texto que envuelve al lector a medida que se mueve a través del espacio, o [Wardrip-Fruin, 2005], donde se trabaja con la lectura como un juego de texto.

Otro género de la literatura digital es la *poesía digital* o *ciberpoesía*. En [Heibach et al, 2004] se indica que la "*ciberpoesía incluye la programación, multimedia, animación, interactividad y la comunicación en red*". Algunos ejemplos de recursos utilizados en poesía digital son el de la *poesía hipertextual* (que usa hipertexto para componer una obra poética), *poesía holográfica*

(poesía creada a través de la técnica holográfica), *poesía creada por ordenador* (poesía generada automáticamente mediante unas reglas lingüísticas o semánticas), poesía que utiliza los caracteres y símbolos ASCII, o *poesía virtual*.

Por último, otra categoría relevante de la literatura electrónica es la representada por el *arte generativo*. En esta categoría se utiliza la implementación de un algoritmo para generar, componer, reordenar o construir textos basados en un esquema aleatorio o en datos que escapan al control del artista. Ejemplos de propuestas, aparte de la poesía creada por ordenador ya aludida anteriormente, son: [Bootz, 2004], donde se describe un generador basado en reglas, que combina sonido con una animación que cambia su ritmo en función de la velocidad de la reproducción del audio, o [Andrews, 2004] donde se interpreta la obra del escritor canadiense [Lionel Kearns] uniendo extractos de dicha obra con imágenes apropiadas.

2.1.3 Ficción Interactiva

Un enfoque de enorme relevancia dentro de la literatura digital es el de la *ficción interactiva*. La ficción interactiva se diferencia del resto de iniciativas mencionadas anteriormente en que contienen elementos de *juego*. Efectivamente, la ficción interactiva no puede llevarse a cabo sin la intervención del lector. [Montfort, 2005] define la ficción interactiva como una colaboración entre el que compone la historia (el autor), el que simula el mundo ficticio en el cual se desarrolla la historia (ordenador), el que explora y modifica el mundo mediante comandos basados en texto (el *interactor*) y el que entiende y contesta a las entradas del *interactor* (programa informático o *parser*).

La ficción interactiva alterna el juego con componentes novelísticos ampliando el repertorio literario. Algunos ejemplos de ello son la obra descrita en [Short, 2002], donde la autora comenta que "*coge los puzles más estándares y aburridos de toda la existencia y fuerza al jugador a resolverlos de una forma diferente con un nuevo sistema*", la presentada en [Ingold, 2001], donde se propone al lector un juego y el objetivo del lector consiste en descifrar el significado de dicho juego después de completarlo, o la descrita en [Leishman, 2004], donde se explora nuevas formas de experimentar una historia, en lugar de esforzarse en resolver rompecabezas y misterios, siendo el objetivo el propio viaje sobre la obra.

2.1.4 Literatura Digital e hipertexto

La literatura digital utiliza habitualmente el *hipertexto* [Conklin, 1987] como un recurso expresivo básico. Así, por ejemplo, el sistema *Storyspace* citado anteriormente se concibió, en realidad, como una herramienta primitiva de edición de hipertextos. A este respecto, este tipo de obras pueden clasificarse en *constructivas* y *exploratorias* [Joyce, 1995]: los hipertextos constructivos son los que construye la audiencia como parte de un proceso de transformación del conocimiento previo, y los hipertextos exploratorios son los que permiten al público guiarse a sí mismos reflejando el sentido de estructura del autor.

De esta forma, un rasgo distintivo de las obras en literatura digital es el de *hipervínculo*. Este concepto permite dotar al lector de la capacidad de elegir qué itinerario seguir durante la lectura del texto, logrando que dicha lectura no sea lineal, sino que, partiendo de un único comienzo, el lector pueda navegar a través del texto sin saber si ya ha leído o no todas las páginas.

2.1.5 Stretchtext

Una propuesta para crear ficción interactiva en un libro digital es el mecanismo del *stretchtext*.

El *stretchtext* fue originariamente propuesto por Nelson en el marco del proyecto Xanadu [Xanadu]. [Brusilovsky P., 1997] define el modelo como un tipo especial de hipertexto. Mientras que el resultado habitual de pulsar sobre una palabra o frase en el hiperenlace provoca moverse a otra página, con el *stretchtext* el texto ni se reemplaza ni se muestra en otra ventana, sino que se presenta (u oculta) la información en la misma página extendiendo el texto. La palabra o frase sobre la que se pulsa se conserva en la pantalla incluso después de que se haya mostrado el texto.

En [Boyle C y Encarnacion A, 1994] se analizan las ventajas del modelo en términos de modelado de usuarios. Efectivamente, un modelo de usuario contiene una representación del conocimiento del lector. Dependiendo de este conocimiento, es posible alterar la cantidad de información presentada ocultando o mostrando el texto. Como consecuencia, el lector ya no

necesita adaptarse al documento, sino que es el documento el que se adapta al lector. De esta forma, el *stretchtext* permite que la información se presente a distintos niveles, lo que permite minimizar la desorientación y maximizar la satisfacción del usuario mediante una transición suave entre los distintos niveles.

En la literatura se puede encontrar numerosos proyectos utilizando *stretchtext* como por ejemplo el de [Brusilovsky et al, 2004] que presenta QuizGuide, un sistema adaptativo desarrollado para ayudar a sus estudiantes a seleccionar los cuestionarios de auto-evaluación más relevantes utilizando la navegación adaptativa. El *stretchtext* se utilizó para mostrar 40 cuestionarios pulsando sobre cada uno de los 20 temas. El objetivo de su utilización consistió en disminuir la sobrecarga de información ocultando los cuestionarios que no se habían realizado. El resultado fue positivo ya que se logró un mayor incremento de sus conocimientos al final del curso. Otro trabajo representativo es el de [Mitsuhara H. et al, 2001], donde se desarrolla un sistema web educativo llamado ITMS (*Individualized Teaching Material System*). Esta aplicación integra el conocimiento distribuido en distintas páginas web y genera material didáctico de varios contenidos de forma individualizada. El *stretchtext* se utiliza para facilitarles una referencia de la información que se va a mostrar cuando pulsen sobre un título o tema. En este caso el autor también afirma que es eficaz para disminuir la sobrecarga de información.

2.2 Herramientas tecnológicas para la literatura digital

Esta sección se dedica a revisar las herramientas tecnológicas que se han utilizado para implementar las ideas e iniciativas propuestas en la literatura digital. Se comienza revisando algunas tecnologías básicas para la representación de contenidos digitales. A continuación, se revisan las tecnologías más relevantes utilizadas en la publicación *web*. Para finalizar, se revisan los dispositivos electrónicos de lectura más extendidos en la actualidad.

2.2.1 Tecnologías Básicas de Representación

La producción de contenidos digitales es una actividad cuyo progreso es dependiente del grado de innovación que se ha conseguido tanto en el hardware como en el software. En un proceso de publicación digital, la organización de los datos y del texto involucrado en las obras

es esencial. Es por ello, que se hace necesario la utilización de una estructura de representación de la información que sea flexible y facilite su reproducción en diferentes dispositivos y con diversos propósitos. En este sentido, los lenguajes de marcado y en general el uso de metainformación son claves para conseguir esta flexibilidad.

Los lenguajes de marcado surgen en el ámbito de la edición digital. En el proceso de composición tipográfica desarrollado durante los años setenta se solía utilizar grandes sistemas comerciales con teclados especiales para etiquetar en los manuscritos aspectos de diseño como encabezados, estilo de fuente y párrafos. El sistema no era totalmente electrónico: así que para comprobar cómo era el resultado final, se tenía que imprimir en papel. En estos primeros sistemas, eran los editores los que marcaban el texto con etiquetas para formatear el texto. No obstante, la composición tipográfica continuó desarrollándose hasta que las pantallas podían mostrar el resultado final una vez que se había finalizado el proceso de composición, sin necesidad de imprimir en papel. Esto permitió involucrar a los propios autores en el proceso.

La inclusión de los autores en el proceso de marcado de los documentos supuso simplificar también los sistemas de marcado, orientándolos a representar la *estructura lógica* de los documentos, en lugar de los aspectos relativos a la tipografía y maquetado final. Para lograr mostrar el resultado final, se utilizaba un programa de formateo que interpretaba las etiquetas y formateaba los contenidos de manera adecuada, de acuerdo a si dichos contenidos se iban a imprimir en papel, o si se iba a generar un producto digital. Esta etiquetación o marcación del documento debía estar adecuadamente formalizada para que todos los intervinientes pudieran trabajar bajo el mismo procedimiento. Por esta razón se desarrolló el lenguaje SGML [Goldfarb, 1991] (*Standard Generalised Mark-up Language*) basado en el lenguaje de marcado general GML (*Generalised Mark-up Language*), creado por IBM.

SGML es un meta-lenguaje, es decir, es una forma de escribir o leer sistemas de codificación. Define la estructura y relación de los datos describiendo su tipo a través de la incrustación de etiquetas. SGML se convirtió en un estándar ISO y fue el predecesor de XML (*Extensible Mark-up Language*) o lenguaje de marcado extensible, [Bray et al, 2008] desarrollado a finales de los

años noventa con la finalidad de ser un meta-lenguaje más flexible, menos complicado de entender y para utilizarse como un lenguaje de transferencia en la web.

XML es un metalenguaje que permite definir lenguajes de marcado o lenguajes de marcas. A partir de estos lenguajes, se crean documentos marcados (instancias del lenguaje de marcas definido) denominados documentos XML. El papel que juega XML en la representación flexible de la información es la posibilidad de separar la representación u organización de la información de su posterior presentación. Se puede tener la información organizada en un documento XML, y a partir de este documento, se pueden tener múltiples presentaciones de la misma información dado que la presentación de la información es algo externo a la propia representación. Otro aspecto relevante con respecto a la flexibilidad de la representación de la información se encuentra en la propia definición de los lenguajes de marcado, dado que estos se pueden crear libremente, adaptados a necesidades particulares, y se pueden extender para cubrir necesidades futuras. Así pues, los documentos XML permiten transportar la información sin ningún tipo de formato, permitiendo, por lo tanto, que estos datos se puedan utilizar en un futuro para presentarlos en diferentes formatos o dispositivos, como, por ejemplo, una página web usando HTML5 o un fichero EPUB.

La estructura de un documento XML está compuesto por: un prólogo (es opcional) cuya misión consiste en añadir información sobre el documento y el cuerpo que contiene la información del documento (la Figura 2.1 muestra un ejemplo simple). El prólogo se puede dividir en dos partes:

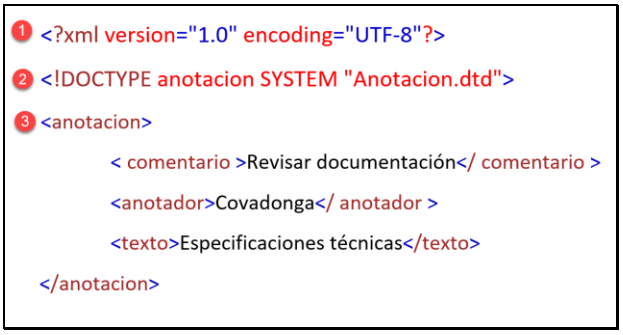


Figura 2.1. Estructura de un documento XML.

- La *declaración* XML (punto 1 de la Figura 2.1): describe las propiedades más genéricas del documento, como la versión del XML utilizada (1.0 en el ejemplo) o la codificación empleada (codificación UTF-8 de UNICODE, en el ejemplo).
- Una DTD (*Document Type Definition*). La DTD es una gramática formal que define la estructura del marcado XML. El procesador XML la utiliza para verificar si el documento es válido. Las DTDs en XML son una simplificación de las DTDs de SGML. Pueden estar definidas dentro de la etiqueta DOCTYPE, ser un documento externo o una combinación de ambas. En la Figura 2.1, se muestra una DTD como una referencia externa. Dicha DTD se muestra en la Figura 2.2:

```
<!ELEMENT anotacion (comentario, anotador, texto)>
<!ELEMENT comentario (#PCDATA)>
<!ELEMENT anotador (#PCDATA)>
<!ELEMENT texto (#PCDATA)>
```

Figura 2.2. Ejemplo de DTD

  - `!ELEMENT anotacion` define que el elemento `anotacion` debe contener elementos "`comentario`, `anotador`, `texto`" en este orden.
  - `!ELEMENT comentario` define el elemento `comentario` como un elemento de *texto* (es decir, que contiene únicamente texto no marcado). Lo mismo ocurre con las definiciones de `anotador` y de `texto`.

El cuerpo, por su parte, contiene los datos a los que se les ha añadido el marcado. Tiene una estructura de árbol. Continuando con la descripción de la Figura 2.1, el cuerpo del documento se identifica como el punto 3 en la figura. La etiqueta `<anotacion>` es la marca del elemento raíz y las tres líneas siguientes describen a los tres hijos del elemento raíz.

Relacionados con los lenguajes de marcados se encuentran los metadatos. Los metadatos son fundamentalmente información o datos sobre otros datos. Describen o definen partes de un documento o el mismo documento sin formar parte de los datos. Su función consiste en hacer que la información sea más manejable y más fácil de encontrar.

La NISO (*National Information Standards Organization*) distingue los siguientes tipos de metadatos [NISO]:

- Metadatos *descriptivos*: Es la información relacionada con el contenido intelectual y que ayuda a encontrarlo o identificarlo.
- Metadatos *administrativos*: Es la información relacionada con la creación del recurso o la información necesaria para administrar dicho recurso. Dentro de este tipo se pueden distinguir:
    - Metadatos *técnicos*: información relacionada con la renderización y decodificación de los archivos digitales.
    - Metadatos de *preservación*: permiten el acceso a los documentos conservados y su gestión para llevar a cabo el proceso preservación digital.
    - Metadatos de *propiedad intelectual*: Detalla los derechos de propiedad intelectual relacionados con el contenido.
- Metadatos *estructurales*: Referidos a la estructura del recurso y a sus elementos.

Los metadatos sólo son útiles si las aplicaciones de software y las personas que lo utilizan los comprenden. Por ello, suelen utilizarse vocabularios o esquemas de metadatos apropiados. Éstos se pueden formalizar a través de organizaciones como la Organización Internacional de Normalización ISO (*International Organization for Standardization*) [ISO], el W3C [W3C], o la NISO [NISO]. La relación que mantienen con los lenguajes de marcado, es que los metadatos pueden modelizarse como un lenguaje de marcado, de manera que los metadatos pueden ser representados como documentos XML. Su aportación principal, con respecto a la representación flexible de la información, se encuentra en la posibilidad de utilizar los metadatos como un conjunto de instrucciones interpretables por los dispositivos de ejecución para mostrar o interaccionar con la información de una forma u otra.

2.2.2 Publicación web

Con la popularización de la WWW desde comienzos del siglo XXI, la *web* se ha convertido en un escenario obligado para la publicación digital. En este escenario, la publicación de las obras digitales se realiza mediante una combinación adecuada de HTML, CSS y JavaScript.

El lenguaje de marcado de hipertexto o HTML es el lenguaje que describe la estructura y el contenido semántico de una página web. Permite organizar el contenido de las páginas web, compartir información, y que los documentos se vinculen y se muestren adecuadamente en el sitio web. Determina el contenido de un documento web pero no su funcionalidad debido a que el lenguaje nació con la intención de compartir información por medio de texto. Este limitado objetivo, motivó a las empresas a desarrollar nuevos lenguajes y programas para añadir nuevas características a la web, como, por ejemplo, funcionalidad (utilizando JavaScript) o definir su estilo (usando CSS). De esta forma:

- La descripción de la funcionalidad de una página web se implementa utilizando un lenguaje interpretado e incluido en los navegadores, llamado JavaScript. Se usa para añadir interactividad dinámica a los sitios web mediante la programación de ciertos comportamientos sobre las páginas web como, por ejemplo, responder a los eventos lanzados por el usuario o crear efectos especiales. Además de proveer a la web de este dinamismo, JavaScript, permite construir aplicaciones web totalmente funcionales.
- Otra tecnología importante son las hojas de estilo en cascada o CSS, utilizadas para describir la presentación o aspecto de la página web. Define el estilo visual de un sitio entero sin necesidad de tener que programar etiqueta a etiqueta cada una de las páginas que lo conforman. Utilizando las CSS se consigue que los archivos no sean demasiado pesados y, además, separan la presentación de la estructura de la página. Las hojas de estilos se pueden configurar para diferentes tipos de representación, como pantallas o impresiones de diferentes tamaños.

Las editoriales han ido cambiando su modelo de negocio porque la evolución y combinación de estas tres tecnologías (HTML, JavaScript y CSS) presentan muchas ventajas frente otras

opciones de publicación actualmente disponibles: el enfoque es compatible con todos los dispositivos, tanto la distribución como la producción es más barata, permite añadir video, imagen, audio, interactividad o animación, la edición del texto se puede realizar de forma sencilla o se pueden añadir fácilmente nuevas librerías o componentes a la publicación.

La descripción de estas tres tecnologías se va a detallar en los siguientes apartados.

2.2.2.1 HTML y HTML5

HTML fue creado por Tim Berners-Lee en 1991 con el objetivo de proporcionar a sus compañeros del CERN (Organización Europea para la Investigación Nuclear) un lenguaje de marcado simple a través del cual pudieran compartir sus trabajos en internet. Este lenguaje permitió que el contenido almacenado en un servidor central se pudiese mostrar en un navegador de un terminal local. Es un estándar basado en el lenguaje de marcado SGML e inicialmente Tim Berners-Lee lo concibió como un derivado suyo. HTML usa el marcado para anotar textos, imágenes, videos y otros contenidos para que se muestren en el navegador web.

La primera versión de HTML (HTML 1.0) surge en 1993 y fue publicada por la IETF (Internet Engineering Tasking Force). Esta versión contaba con algunas etiquetas y gráficos básicos.

En el año 1995, surge el primer estándar oficial de HTML, la versión HTML 2.0. Se simplifica su estructura para que la edición fuese más ágil. Se introducen los formularios con un conjunto limitado de elementos y botones de opción.

En el año 1996, el consorcio sin ánimo de lucro W3C (*World Wide Web Consortium*) se encarga de las especificaciones de HTML. En esa época ya había surgido la versión 3.0, pero nunca se estandariza.

En 1997, se publica HTML 3.2, siendo la primera recomendación de HTML publicada por el W3C. Introdujo las CSS de nivel 1 aunque los navegadores tardaron en adoptar esta nueva forma de dar formato. También introdujo las tablas, figuras, marcos, ecuaciones matemáticas, *applets* de Java y texto que fluye alrededor de las imágenes.

HTML 4.0, se publicó en 1998, introduciendo una nueva forma de implementar el diseño web animando a los diseñadores/programadores a mover los parámetros de formatos a las hojas de

estilo para fomentar que los cambios de formato se realizasen dentro de una hoja CSS en lugar de editar cada una de las páginas web contenidas en el sitio web. Incluyó también, características de accesibilidad, mejoras en tablas y formularios, o la inclusión de scripts en las páginas web.

En 2004, compañías como Apple, Mozilla y Opera, ante la falta de iniciativa de W3C en referencia a HTML forman una asociación llamada WHATWG (Web Hypertext Application Technology Working Group) [WHATWG] y desarrollan HTML5 como el nuevo lenguaje estándar. En 2007, W3C retomó el proyecto de HTML5 y se unió al trabajo de WHATWG.

[Gauchat, 2012] define a HTML5 como un nuevo concepto de construcción de sitios web y aplicaciones para dispositivos móviles, trabajos en red o computación en la nube. La comunidad WHATWG detalla todas las mejoras de esta nueva versión. En, particular, para este trabajo son especialmente relevantes las siguientes:

- Incluye nuevas etiquetas que permiten la creación de una estructura para la web semántica. Al dotar de contenido semántico a las páginas pueden almacenar, entender e interpretar la información. HTML5 [Faulkner et al, 2016] ha introducido 23 etiquetas nuevas para promover la búsqueda en un contexto de web semántica.
- Incluye nuevas interfaces de programación de aplicaciones o APIs (*Application Programming Interface*) que permiten hacer uso de funciones o infraestructura existente en otros programas, como, por ejemplo:
    - Nuevos controles para reproducir y sincronizar elementos multimedia.
    - Arrastrar y soltar elementos, característica que consiste en arrastrar un objeto y moverlo a otra ubicación. Cualquier elemento puede tener esta característica.
    - Imprimir el contenido de la ventana actual abriendo un cuadro de diálogo para permitir al usuario seleccionar las opciones de impresión.
    - Ejecutar scripts en segundo plano sin afectar al rendimiento de la página web.

    - Almacenamiento en el lado cliente. Las aplicaciones web pueden almacenar datos localmente en el navegador del cliente, sin depender del mecanismo de *cookies*.
    - Geolocalización: Función que devuelve la posición geográfica del usuario.
- Permite que los elementos MathML o lenguaje de marcado matemático y los gráficos vectoriales o SVG, se utilicen dentro de un documento web.
- Se introducen nuevos elementos, permitiendo una estructura más organizada, como, por ejemplo:
    - `Footer`: pie de página de una sección.
    - `Nav`: define un conjunto de enlaces de navegación.
    - `Dialog`: para representar conversaciones.
    - `Figure`: especifica contenido autónomo como diagramas o ilustraciones.
    - `Embed`: define un contenedor para una aplicación externa o plug-in.
    - `Canvas`: se utiliza para dibujar gráficos a través de JavaScript.
- Se añaden nuevos atributos, algunos de ellos son:
    - El elemento `table` tiene ahora un atributo `sortable` y el elemento `th` un atributo `sorted`, para ordenar las columnas de una tabla.
    - `Contextmenu`: especifica un menú contextual para un elemento.
    - `Contenteditable`: especifica si el contenido de un elemento es editable o no.

Todas estas mejoras en HTML5 facilitan el trabajo de los editores puesto que es independiente del dispositivo en que se visualiza, se puede insertar video o audio sin necesidad de depender de plug-ins externos para reproducirlos, incorpora funciones de geolocalización, tiene más flexibilidad para crear controles de formularios o permite una estructuración de la página más afinada.

2.2.2.2 CSS y CSS3

Las hojas de estilo CSS (*Cascading Style Sheets*) es una tecnología creada por el W3C con la finalidad de separar formalmente la estructura de la página web de la presentación. Se diseñaron con el fin de mover los parámetros relacionados con los formatos ubicados en las etiquetas HTML a las hojas de estilo para que los cambios de formato se realizasen dentro de una hoja CSS en lugar de editar cada una de las páginas web contenidas en el sitio web. De esta forma, cuando se necesita realizar un cambio, simplemente se actualiza la CSS y el cambio se ve reflejado en todos los documentos que la utilizan.

Los navegadores web aplican las reglas CSS a los documentos que la utilizan para determinar el formato. Las reglas CSS están formadas por un conjunto de *propiedades* que contienen los valores de los atributos de formato a aplicar, y un *selector*, que permite determinar el elemento o elementos sobre los que debe aplicarse la regla.

En 1996, el W3C publicó la primera recomendación oficial, conocida como CSS1. Esta versión se centraba en la apariencia, introduciendo, por ejemplo, algunas propiedades y atributos básicos relacionadas con las fuentes, color de texto, fondos o alineaciones.

En 1998, se publica la segunda recomendación oficial, llamada "CSS nivel 2". Añade capacidades de posicionamiento para texto y objetos, incorpora funcionalidades de capas, el concepto *media types* (presentación del documento en diferentes soportes) o soporte para las hojas de estilo auditivas.

A mediados de 1999 se liberan los primeros borradores de CSS3. A diferencia de las versiones o generaciones anteriores, CSS3 está dividida en varios módulos separados [Kesteren et al, 2014]. Cada módulo añade nuevas funciones a las definidas en las versiones anteriores previas y es compatible con todo tipo de plataformas y dispositivos. Esta división modular facilita el mantenimiento porque los cambios se realizan sobre cada uno de los módulos sin afectar a otros componentes. Algunas de las características o módulos más destacables de esta nueva versión son:

- *Selectores*: Permite a los desarrolladores editar los elementos por nombre, clase, tipo, atributo, etc. Incluye los selectores de CSS2.1 y añade decenas de

selectores, seudo-clases y seudo-elementos. La lista completa se puede consultar en [W3C]

- *Fondo y bordes*: incluye características para redondear esquinas, poner una imagen como borde, degradar la imagen de fondo o incluir múltiples imágenes de fondo.
- *Transformaciones en dos y tres dimensiones*: Permite trasladar, rotar, escalar e inclinar elementos.
- *Animaciones*: Permite la animación de la mayoría de elementos HTML sin necesidad de usar Flash o JavaScript, simplemente cambiando gradualmente de un estilo a otro. De esta forma se consigue que el tiempo de carga de la página sea inferior a sus precursores.
- *Interfaz de usuario*: tiene nuevas características de interfaz de usuario como elementos para redimensionar, contornos y tamaños de caja.
- *Media query*: Permite que la presentación del contenido se adapte a los dispositivos de salida sin tener que cambiar el contenido.

La implementación del aspecto de una página web se puede realizar utilizando el lenguaje CSS o mediante la utilización de un preprocesador de hojas de estilo, como por ejemplo [Saas] (Syntactically Awesome Stylesheets). SaaS optimiza el desarrollo porque es modular (permite crear varios archivos CSS y compilarlos todos en un solo CSS), permite la herencia para extender clases, el uso de variables y funciones para evitar repetir código, la reutilización a través de los *mixins* (bloques de código reutilizables), la importación de archivos o clases, la jerarquización y organización del código a través de la creación de la anidación de selectores. Otros preprocesadores interesantes son el preprocesador Less [Less], utilizado por el núcleo de Bootstrap (ha sido desarrollado a partir de CSS por lo que utilizan una sintaxis muy similar), o Stylus [Stylus], parecido a Less pero con una sintaxis más concisa.

2.2.2.3 Dinamismo y JavaScript

JavaScript es un lenguaje de programación interpretado, creado por Brendan Eich en 1995, que surgió para cubrir la necesidad de ampliar las posibilidades de HTML. Es independiente de la plataforma y compatible con la mayoría de navegadores web.

Se usa para añadir interactividad dinámica a los sitios web mediante la programación de ciertos comportamientos sobre las páginas como, por ejemplo, responder a los eventos lanzados por el usuario o crear efectos especiales. Además de proveer a la web de este dinamismo, JavaScript, permite construir aplicaciones web totalmente funcionales.

El navegador web del cliente es el que se encarga de interpretar las instrucciones de JavaScript, es decir, el código se ejecuta en el lado del cliente en lugar de en el servidor ahorrando ancho de banda, reduciendo la carga del lado del servidor y ejecutándose más rápido porque lo hace inmediatamente, en lugar de contactar con el servidor y esperar su respuesta.

En sus primeras versiones, JavaScript se limitó a interactuar con los formularios y el usuario, además de detectar cuando éste lanzaba determinados eventos.

Cuando los navegadores web comenzaron a ser compatibles con la estructura de objetos DOM (*Document Object Model*), el poder de JavaScript se incrementó, porque este avance le permitía alterar dinámicamente el aspecto, el contenido y los atributos de una página.

JavaScript ha ido evolucionando a largo de todos estos años incorporando mejoras como, por ejemplo, expresiones regulares, manejo de excepciones con try/catch, bibliotecas para soporte de JSON (*JavaScript Object Notation*), generadores de expresiones estilo Python o *pattern matching* (definición de funciones por medio de la correspondencia de parámetros y resultados).

La revolución de JavaScript llega con la creación de HTML5, puesto que más de la mitad de las características de HTML5 son APIs de JavaScript. HTML5 define APIs para, por ejemplo, trabajar con el GPS, la cámara, la pantalla, con el navegador, para procesar o sintetizar audio en las aplicaciones web, para insertar audio y video en una web y controlar su reproducción a través de la creación de controles personalizados o extraer información adicional de un video.

Otros avances tecnológicos relevantes en el ámbito del dinamismo o comportamiento de los libros digitales creados utilizando la tecnología web, son los que se van a detallar en las subsecciones siguientes.

2.2.2.3.1 Componentes web

Los *Web Components* o componentes web es el estándar de la W3C enfocado en la creación de etiquetas HTML personalizadas que encapsulan marcado, código JavaScript y estilos CSS, permitiendo su reutilización. Las especificaciones son las siguientes [Glazkov e Ito, 2014]:

- *HTML Templates*: Define cómo declarar un fragmento de contenido del lado cliente que no se usa durante la carga de la página para poder instanciarlo posteriormente en tiempo de ejecución. El parser sólo valida si el contenido es correcto mientras carga la página, no lo renderiza, reduciendo, por lo tanto, las solicitudes iniciales de la carga de la página. El contenido de las plantillas no forma parte del DOM, permaneciendo oculto para el navegador hasta que se instancie. Esta ocultación acelera las búsquedas porque JavaScript recorre el DOM sin incluir estas plantillas ocultas.
- *HTML Imports*: La especificación *imports* define la inclusión y reutilización de documentos HTML en otros documentos HTML. *Imports* incluye cualquier recurso utilizado dentro del documento importado, como otros componentes, código JavaScript, CSS o imágenes. HTML incluye un mecanismo llamado *de-puting* que impide múltiples solicitudes a la misma URL de recursos, permitiendo, por ejemplo, que, si varios recursos importados hacen referencia a la misma dirección web, el navegador sólo recupere el recurso una vez.
- *Shadow DOM*: Proporciona la encapsulación para el DOM y las CSS en un componente web. Es una técnica que incluye un subárbol de elementos en un documento, pero no en el árbol DOM del documento. El concepto es similar a los *iframes,* pero, en lugar de insertar un nuevo documento, forma parte del documento principal.

- *Custom Elements*: Permite crear elementos HTML personalizados añadiendo comportamiento JavaScript y hojas de estilo CSS además de proporcionar comportamientos en las distintas etapas del ciclo de vida del nuevo elemento.

2.2.2.3.2 Frameworks y bibliotecas JavaScript

Las innovaciones incorporadas en la versión 5 de HTML junto con los nuevos paradigmas de programación han impulsado la proliferación de multitud de bibliotecas y entornos de trabajo o *frameworks*, que facilitan y agilizan el trabajo de los desarrolladores, permite la reutilización del código, son compatibles con la mayoría de los navegadores y están en constante evolución, corrigiendo *bugs* y lanzando nuevas versiones.

En esta subsección se van a comentar algunos de los *frameworks* o bibliotecas más útiles en el proceso de edición de una obra digital.

2.2.2.3.2.1 Frameworks MVC

El paradigma de programación *Modelo Vista Controlador* (MVC) separa el código de las aplicaciones por responsabilidades facilitando el mantenimiento y la organización y la reutilización del código. Los modelos se encargan de la lógica de negocio, las vistas con la lógica de presentación y los controladores conectan las vistas y los controladores.

Existen numerosos *frameworks* MVC para JavaScript. Algunos de los más populares son los siguientes:

- Backbonejs [Backbonejs]: Framework ligero y sencillo, diseñado para desarrollar aplicaciones web de una sola página. Además, es flexible puesto que es compatible con otros frameworks.
- AngularJS [AngularJS]: Es un framework MVW o Modelo Vista cualquier paradigma, lo que da flexibilidad sobre el patrón de diseño a utilizar. Diseñado y mantenido por Google. El enlace de datos es bidireccional permitiendo la sincronización automática de datos entre el modelo y la vista dejando que el desarrollador únicamente se preocupe de los datos. El código está en módulos favoreciendo la reutilización, el mantenimiento y las pruebas.

- EmberJS [EmberJS]: Framework con una sólida capa de datos que permite su integración con APIs de servicios REST proporcionando la información en formato JSON. La velocidad de carga y ejecución de la aplicación son bastante rápidas.

2.2.2.3.2.2 Frameworks CSS

Estos *frameworks* están compuestos por bibliotecas de estilos genéricos con el propósito de acelerar el proceso de maquetación de páginas web. Se componen de código HTML para formar la estructura de la página, diferentes estilos tipográficos, y hojas de estilo CSS, JavaScript para cambiar dinámicamente elementos, como, por ejemplo, menús desplegables y compatibilidad entre navegadores. A continuación, se va a comentar algunos de los frameworks de código abierto más populares:

- *Bootstrap* [Bootstrap]: Creado por Twitter. Se caracteriza porque resulta muy sencillo maquetar páginas web con sus clases CSS y los diseños visuales de los principales elementos HTML. Es *responsive* o adaptativo, es decir, se visualiza correctamente en distintos dispositivos.
- *Semantic UI* [Semantic UI]: Su funcionamiento es similar a Bootstrap. Contiene más tres mil temas y 50 componentes prediseñados. Está integrado con otros frameworks como Angular, React js [React js] (biblioteca de creación de interfaces de usuario creada por Facebook), Meteor js [Meteor js] (*framework* para automatizar y simplificar el desarrollo de aplicaciones web con JavaScript) o EmberJS. Las clases utilizan sintaxis de lenguajes naturales para vincular conceptos de forma intuitiva.

Otros frameworks CSS destacables son: [Foundation] provee un entorno para la personalización de sitios web con características base por defecto o [Skeleton] framework con numerosas opciones de personalización.

2.2.2.3.2.3 Animaciones, juegos y experiencias de realidad virtual

Algunas de las bibliotecas JavaScript de código abierto más utilizadas para la generación de experiencias interactivas avanzadas son las siguientes:

- Three.js [Three.js]. Biblioteca ligera creada en el año 2009. Genera y anima gráficos en tres dimensiones dentro de un navegador. Aprovecha las posibilidades de HTML5 siendo capaz de generar escenas en tres dimensiones con WebGL, en dos dimensiones con Canvas e imágenes vectoriales con SVG. Permite importar archivos en 3D a partir de las aplicaciones Blender o Maya. Incorpora shaders (utilizados para crear transformaciones y efectos especiales) que se pueden personalizar con el lenguaje GLSSL (OpenGL Shading Language).
- Babylon [Babylon]. Creado por Microsoft en 2013. Genera juegos en 3D con HTML5, WebGL y Web Audio. Contiene motores de colisiones, de física, audio y optimización.
- A-Frame [A-Frame]. Es un *framework* utilizado para construir experiencias de realidad virtual WebVR utilizando elementos HTML personalizados, construido por Mozilla. WebVR es una API de JavaScript creada también por Mozilla, para capturar la información de los dispositivos de realidad virtual conectados a un ordenador y convertirla en datos útiles para que los juegos u otras aplicaciones lo usen. Estos elementos HTML personalizados usan Three.js y WebGL para crear elementos habilitados para la realidad virtual en una escena, sin necesidad de que los desarrolladores tengan que aprender WebGL.
- React VR [React VR]. Es un *framework* creado por Facebook, utilizado para la construcción de experiencias de realidad virtual. Aprovecha también las APIs de WebGL y WebVR. Permite la utilización de los componentes React para componer escenas en 3D, combinándolas con interfaces de usuario panorámicas de 360º, texto e imágenes.

Otras bibliotecas interesantes para generar animaciones y juegos son: Playcanvas [Playcanvas], diseñada exclusivamente para la creación de juegos, o WhitestormJS

[WhitestormJS], que soporta la simulación de "Soft Body" o simulación de un objeto deformable y que, además, integra la librería Three.js.

2.2.2.3.2.4 *Frameworks* y bibliotecas de visualización de datos

Este tipo de *frameworks* y bibliotecas se encargan de la presentación de los datos en diagramas o gráficos dinámicos. Algunos ejemplos de código abierto son los que se listan a continuación:

- D3.js [D3.js]: Crea una gran variedad de visualizaciones de datos dinámicas con transiciones, iteraciones y transformaciones usando HTML5, SVG y CSS.
- Chart.js [Chart.js] Visualiza datos usando el elemento Canvas de HTML5 para dibujar gráficos y tablas.

Otras interesantes utilidades gráficas son: Google Charts [Google Charts], API de Google para la visualización de gráficos representados mediante HTML5 y SVG para garantizar la compatibilidad entre navegadores, o Chartist.js [Chartist.js], para crear gráficos escalables y responsivos.

2.2.2.3.2.5 *Bibliotecas* de maquetación de libros digitales

Se encargan de crear efectos de desplazamiento de páginas y añaden contenido interactivo para simular un libro impreso en forma de libro digital. Algunas librerías de código abierto interesantes son:

- Turn.js [Turn.js]. Biblioteca responsiva y sencilla de usar. Contiene las características de *zoom* y cortar una página en dos partes.
- BookBlock [BookBlock]. Permite la incorporación de video, audio, imágenes vectoriales e hiperenlaces.
- Booklet [Booklet]. Biblioteca sencilla, rápida y ligera.

Otra opción para crear libros electrónicos, es la utilización de plataformas web que, además de proporcionar herramientas para la generación también permiten alojar y compartir publicaciones interactivas de forma gratuita, como son Calameo [calameo], Joomag [joomag] o Yumpu [Yumpu].

2.2.3 Publicación en Dispositivos

En esta sección se hablará sobre los *ebooks* (libros electrónicos) y los dispositivos dedicados a la lectura de libros electrónicos o e-readers.

2.2.3.1 Libros electrónicos

[Morgan, 1999] define los ebooks, como una combinación de hardware y software diseñada para la lectura, en contraste con los textos electrónicos que se pueden leer en un ordenador.

La primera propuesta de lector mecánico, precursor de los libros electrónicos, lo inventó una maestra de Ferrol llamada Ángela Ruiz Robles en el año 1949. Ángela Ruiz tuvo esta idea innovadora al ver todos los días a sus alumnos cargar con los libros de texto con el problema de salud que suponía cargar con pesadas carteras. Así que creó su enciclopedia mecánica para compactar todos los libros de texto en uno menos pesado.

En 1971, Michael S. Hart crea el primer libro electrónico al digitalizar la Declaración de la Independencia de los Estados Unidos. Unos años más tarde, en 1987, la empresa East Gate System, publica y distribuye en disquete, el primer libro de hipertexto [Joyce, 1987]. Esta obra se ofreció originalmente como una demostración de Storyspace (véase la sección 2.1).

A principios de los años 2000, los libros digitales en formato PDF y los primeros formatos de ebook se utilizaban en ordenadores de sobremesa y portátiles, pero este uso no coincidía con las preferencias de los lectores del libro impreso ya que éstos buscaban un dispositivo portable, más liviano, fácil de usar y más barato.

Por esta razón, se comenzó a desarrollar hardware para implementar dispositivos portables dedicados a la lectura de libros. En 2004 Sony lanza el primer lector ebook que utiliza la tecnología *eInk* para evitar el resplandor de las pantallas retroiluminadas, y desarrolló el Sony e-reader en el 2006. En el año 2007, Amazon irrumpe en el mercado lanzando su primera versión de *Kindle*, logrando un éxito notable aprovechando que contaba con una gran base de datos de clientes de libros electrónicos, una configuración sencilla y acceso a una amplia biblioteca de *ebooks*.

Estos nuevos e-readers se desarrollaron con una serie de características que impulsaron su popularidad: por ejemplo, ser dispositivos portables, contar con una batería de larga duración, una capacidad de memoria suficiente como para almacenar una gran cantidad de libros a la vez, o la utilización de tecnologías que eliminan el resplandor de las pantallas retroiluminadas.

Como alternativa a los *e-readers* surge un importante segmento de mercado orientado a los *smartphones* y a las tabletas. Este mercado tiene múltiples propósitos y uno de ellos es permitir descargar y leer libros mientras que los e-readers sólo tienen uno. De esta forma, en el año 2008, la tienda *online Books on Board* comienza a vender ebooks para iPhones. Un año más tarde, en 2009, se unen grandes editores (Condé Nast, Hearst, Meredith, News Corp y Time Inc.) para desarrollar un formato de libro electrónico y una tienda online para para competir con Amazon por una posición dominante del mercado. En el año 2010, el *iBookStore* de Apple pone en venta *ebooks* para IPad, y un año más tarde Google, a través del Google Market y la aplicación de Google Books, ofrece libros electrónicos para visualizarlos en cualquier dispositivo que disponga de un sistema operativo Android instalado.

2.2.3.2 Formatos para libros electrónicos

Un aspecto fundamental para el soporte de libros electrónicos es la definición de formatos que garanticen la interoperabilidad entre distintos dispositivos. A continuación, se revisan los más relevantes.

2.2.3.2.1 OEBPS

A finales de la década de los noventa se funda un consorcio (*Open Ebook Authoring Group*) durante la primera conferencia dedicada a los libros electrónicos, para desarrollar un formato de *ebook* abierto con el propósito de crear un documento fuente que se pudiese leer en diferentes dispositivos y aplicaciones software. La independencia con la plataforma era fundamental ya que el estándar que tenían como referencia en ese momento era el PDF, formato inadecuado para mostrar los *ebooks* en dispositivos móviles como la BlackBerry.

En 1999, se renombra el grupo por *Open Ebook Forum* y el primer producto surgido del grupo fue el OEBPS 1.0 (*Open eBook Publication Structure*). En 2002 se publicó la versión 1.2 en la que

se separaba el contenido del diseño mediante el uso de hojas de estilo CSS. Ese documento fuente consistía en documentos XHTML, CSS, multimedia y un esquema XML para listar los componentes del libro electrónico. En 2010 se publica una actualización en la que se divide el documento en tres partes: el OPS (Open Publication Structure) en el que se describe el formato del contenido, el OPF (acrónimo del inglés Open Packaging Format) que define la estructura del documento XML y el OCF (del inglés Open Container Format) que comprime los archivos como Zip.

2.2.3.2.2 MOBI

En 2005 el *Open Ebook Forum* cambia su nombre por IDPF [IDPF] (*International Digital Publishing*). Empiezan a percatarse de que el OEBPS 1.2 carecía del apoyo en la distribución ya que carecía de derechos de autor y, además, faltaba interactividad con los usuarios finales. Por ello, las empresas estaban trabajando en crear su propio estándar utilizando OEBPS en el lado de la producción, pero extendieron el estándar en la distribución con compresión o códigos binarios propietarios, como por ejemplo el formato MOBI de la empresa francesa *MobiPocket*, adquirida por Amazon en 2005. Para los *ebooks* protegidos por *copyright*, Amazon usa el formato AZW que es una variante del formato MOBI, pero con una compresión adicional.

El formato MOBI está basado en el formato OEBPS y puede incluir JavaScript, *frames*, marcadores y la posibilidad de añadir notas. Está compuesto por un archivo HTML utilizado como base, un fichero KF7 que es un archivo MOBI *Kindle* optimizado para que los dispositivos más antiguos lo puedan leer y, finalmente, un archivo MOBI "KF8" que integra las características de EPUB 3 (HTML5 y CSS3) proveyendo al libro electrónico de más estilos, estructura y funcionalidad.

2.2.3.2.3 EPUB

La proliferación de formatos propietarios no coincidía con el objetivo de IDPF que pretendía definir un estándar que abarcase tanto la producción como la distribución. Una aproximación a este ideal es el formato EPUB, formato que se adoptará en este trabajo para la publicación para dispositivos.

EPUB o ePUB (Electronic Publication), es un formato de distribución e intercambio de publicaciones digitales y documentos basados en estándares web creado por IDPF [IDPF]. Tiene como una de sus principales ventajas que se adapta a cualquier dispositivo de lectura.

EPUB versión 2, se convirtió en un estándar en septiembre de 2007 y el sucesor del formato OEBPS.

EPUB versión 3, reemplazó a la versión 2 en octubre de 2011 al ser aprobada como especificación recomendada. EPUB3 consiste, en realidad, en cuatro especificaciones diferentes:

- *Publicaciones*: define la semántica y los requisitos generales.
- *Documentos de contenido*: define el formato de contenido.
- OCF: define un formato de archivo y el contenedor en formato Zip.
- *Media Overlays*: define un formato y un modelo de procesamiento para la sincronización de texto y audio.

EPUB3 Incorpora las siguientes tecnologías convirtiéndolo en un formato más versátil [Garris, 2011]:

- XHTML5 para representar el contenido de texto y multimedia.
- SVG 1.1 para la representación de trabajos gráficos.
- CSS 2.1 y 3 para la modificación del aspecto y el renderizado del contenido.
- JavaScript para darle interactividad a la publicación.
- TrueType y WOFF para dar soporte de más tipos de fuentes.
- SSML/PLS/CSS 3 Speech para mejorar el renderizado de texto a voz.
- SMIL 3 para sincronizar la reproducción de texto y audio.
- La incorporación de MathML a HTML5 permite el uso de expresiones matemáticas en los *ebooks*.
- Vocabulario RDF para permitir embeber información semántica sobre la publicación y su contenido.

- XML para facilitar aspectos de procesamiento de las publicaciones.
- ZIP para comprimir los recursos en un único fichero.
- Al está basado en HTML5 permite añadir elementos de audio y video embebiéndolos directamente en el documento.

EPUB versión 3.1, es la última versión, publicada en enero de 2017.

2.2.3.2.3.1 Estructura de un fichero EPUB

Un fichero con extensión. epub es un fichero comprimido con la siguiente estructura:

- *Fichero OPF:* Define la estructura de los documentos que contiene un libro. Consta de dos partes:
    - *Manifiesto*: contiene una lista detallada con todos los recursos que constituyen el libro. La lista contiene los siguientes elementos:
        - un id,
        - href: contiene el nombre y extensión del fichero
        - media-type, contiene el tipo de recurso.
    - *Spine*: Define el orden de lectura del libro. Es una lista ordenada de los recursos de publicación enumerados en el manifiesto.
- *Metadata*: Información relacionada con el propio libro. Es opcional.
- *Recursos*: todos los ficheros que estén listados en el manifiesto.

2.3 A Modo de Conclusión

En este capítulo se ha realizado una introducción a la literatura digital y a algunas de sus iniciativas. Así mismo, se ha analizado la capa tecnológica que se ha utilizado para implementarla. En este sentido, se ha puesto de manifiesto la importancia del uso de XML y de la tecnología web en el campo de la producción de contenidos. Los documentos XML ofrecen una estructura flexible que permite crear documentos para transportar la información sin ningún tipo de formato, permitiendo que los datos se puedan utilizar en un futuro para presentarlos en cualquier formato de salida. Por otro lado, la tecnología web con la aparición de la versión 5 de HTML facilita el trabajo de los editores puesto que es independiente del dispositivo en que se visualiza, se puede insertar video o audio sin necesidad de depender de plug-ins externos para reproducirlos, incorpora funciones de geolocalización, tiene más flexibilidad para crear controles de formularios o permite una estructuración de la página más afinada.

También se ha analizado el cambio que se ha producido en la publicación de libros con la aparición de los libros digitales. Ya no es necesario producir productos físicos e incluso electrónicos, como el CD-ROM, para poder distribuir las obras, sino que el producto se puede entregar a través de internet con libros en formato web o formato de dispositivos electrónicos (EPUB o MOBI, por ejemplo).

Por otro lado, se ha comentado el impacto que ha producido la aparición y desarrollo de dispositivos electrónicos como los e-readers, *smartphones* y tabletas que con características como, por ejemplo, ser dispositivos portables, contar con una batería de larga duración, disponer de una capacidad de memoria suficiente como para almacenar una gran cantidad de libros a la vez o la utilización de tecnologías que eliminan el resplandor de las pantallas retroiluminadas, han impulsado las ventas de libros electrónicos.

Todas estas innovaciones han cambiado el modelo de negocio de las editoriales y de los escritores porque el material en formato digital reduce el riesgo de mantener grandes cantidades de existencias y permite, además, que su impresión sea fácil y rápida. Los autores

pueden imprimir un número reducido de libros, lo que contribuye a impulsar a la industria de la autoedición.

El presente trabajo se enmarca en este contexto, abordando el problema de la creación sencilla de obras digitales de ficción interactiva basadas en el paradigma de texto extensible, así como en su publicación, tanto en entornos *web*, como sobre dispositivos electrónicos. La principal problemática abordada en este trabajo es doble. Por una parte, es necesario facilitar a los autores, autores que normalmente no poseen conocimientos avanzado sobre tecnologías *web* o formatos de libros electrónicos, la producción de estas obras. Por otra parte, es necesario facilitar la publicación de las obras en distintas plataformas (tanto para la *web*, como para dispositivos). En ambos casos será necesario abordar las características de interactividad inherentes a este tipo de géneros literarios. En los siguientes capítulos se describe la solución inicial planteada, así como la evaluación inicial de su usabilidad.

Capítulo 3 - La herramienta ILSAditor

En este capítulo se presenta la especificación de requisitos de la herramienta desarrollada en el presente trabajo fin de master, así como su arquitectura e implementación.

3.1 Requisitos generales de la herramienta

Los requisitos generales planteados para la herramienta eran:

- Implementar un editor que permita la creación, edición y publicación de libros digitales.
- Facilitar la creación de libros digitales cuyos contenidos se puedan organizar de acuerdo al paradigma *stretchtext.*
- Ofrecer una interface de creación y de lectura de textos de tipo WYSIWYG con iconos intuitivos para aplicar los efectos de código necesario y que no requiera contar con unos conocimientos profundos de informática. De esta manera, la edición del documento se realizaría viendo directamente el resultado final.
- Permitir la exportación del contenido editado a un formato orientado a web (HTML5, JavaScript y CSS) o a un formato orientado a dispositivos electrónicos (formato EPUB).

3.2 Requisitos funcionales de la herramienta

A continuación, se van a revisar los requisitos funcionales de la herramienta ILSAditor.

3.2.1 Carga del libro digital

El texto que se editará en la herramienta se deberá poder introducir en el editor escribiendo directamente en el mismo o bien abriendo un archivo existente en formato de texto plano, HTML o EPUB.

En el caso de usar la opción de abrir un archivo existente, la herramienta proporcionará una barra de herramientas con una opción de apertura de archivo. Al pulsar sobre dicha opción, se abrirá una ventana para permitir seleccionar el archivo con el que se desea trabajar. Una vez seleccionado, se abrirá el contenido del mismo en el editor.

3.2.2 Insertar imagen de fondo

Esta funcionalidad facilitará al usuario agregar una imagen de fondo al contenido. Para ello, la herramienta proporcionará un botón que, al ser pulsado, mostrará una ventana en la cual el usuario podrá elegir la imagen de fondo que quiere mostrar y establecer las dimensiones de la misma. La imagen de fondo elegida deberá mostrarse una vez que el libro digital se haya guardado y generado.

3.2.3 Edición del stretchtext

Esta funcionalidad estará orientada a facilitar la estructuración del contenido del libro digital usando *stretchtext*. Para ello, la herramienta deberá permitir seleccionar un trozo de texto del libro digital, y facilitar un botón en la barra de herramientas que, al ser pulsado, generará como resultado la definición de un nivel de estructuración jerárquico (o nivel de *stretchtext)*. Una vez definido, el texto deberá ser resaltado visualmente y el usuario deberá introducir un nombre para el nivel definido. Como resultado de la definición del nivel, la herramienta deberá permitir ocultar o develar el texto, al pulsar con el ratón sobre dicho texto.

La herramienta deberá permitir crear diferentes niveles de estructuración. En este sentido, para crear otros niveles, siempre se tendrá que hacer dentro del nivel anterior al que se desea añadir el *stretchtext*. Con el objetivo de distinguir los distintos niveles definidos, el editor deberá aplicar un formato personalizado al texto que pertenece a uno determinado.

3.2.4 Eliminación de texto stretchtext

Para eliminar un nivel de *stretchtext* que se ha definido sobre un texto, la herramienta proporcionará un botón que al ser pulsado eliminará el nivel de *stretchtext* definido sobre un texto. Para ello, previamente se deberá seleccionar el texto asociado al nivel de *stretchtext* que se desea eliminar.

3.2.5 Insertar imágenes en un nivel de stretchtext

La herramienta deberá permitir añadir una imagen al principio de un nivel de *stretchtext*. Para ello, se proporcionará un botón de inserción de imágenes. Previamente, se habrá elegido con el ratón el lugar de inserción de la imagen. Al pulsar el botón de inserción, se abrirá una

ventana para permitir seleccionar un archivo de imagen, añadir una descripción, un espacio vertical y horizontal, así como indicar su posición en relación al texto. Además, se deberá permitir enlazar la imagen con una URL, para que, una vez se genere el documento, el usuario pueda navegar a otra página pulsando sobre la imagen. Una vez que se haya añadido la ruta de la imagen y todas las propiedades que necesite, se deberá abrir la imagen en el editor.

3.2.6 Insertar audio en un nivel de stretchtext

El editor permitirá añadir audios para que se reproduzcan al pulsar sobre un texto con un nivel de *stretchtext* definido, un botón o una imagen.

Para insertar un archivo de audio, la herramienta proporcionará un botón para añadir el audio, que al ser pulsado lanzará una ventana que permitirá al usuario seleccionar el archivo de audio que se desea reproducir. Previamente, el usuario deberá haber seleccionado un texto con un nivel de *stretchtext* definido, un botón o una imagen.

3.2.7 Insertar enlace

Esta funcionalidad permitirá agregar un enlace al contenido y editar las propiedades del enlace. La herramienta facilitará un botón para este propósito. En primer lugar, el usuario deberá seleccionar el texto al que se quiere añadir el enlace. A continuación, se deberá pulsar sobre el botón de inserción, el cual abrirá una ventana en la que se podrá optar por una de las siguientes opciones:

- Introducir manualmente la URL con la que se quiere vincular el texto.
- Seleccionar un documento HTML del sistema de ficheros local. Si se selecciona esta opción, se abrirá una ventana para permitir seleccionar una página web.
- Crear un documento HTML vacío. Se abrirá una ventana para seleccionar el directorio y el nombre del documento HTML a crear. Una vez introducido el nombre, se deberá pulsar sobre un botón guardar que creará un documento HMTL vacío sobre el que se vinculará el texto seleccionado.

3.2.8 Cambiar el tipo de maquetación a fija

Esta funcionalidad estará orientada a facilitar el diseño de libros digitales destinados a publicarse en tabletas con una pantalla de dimensiones 768x1024 píxeles. Los libros generados mediante esta opción tendrán la dimensión especificada y no se modificarán al visualizarse en otros dispositivos. En el editor, el área de texto se deberá indicar de alguna forma visual, y se forzará al usuario a editar su obra dentro de esta área.

3.2.9 Creación de anotaciones

Esta funcionalidad deberá permitir la inserción de anotaciones en forma de ventanas emergentes con contenido multimedia. El objetivo será añadir información adicional a un trozo de contenido textual seleccionado previamente por el usuario. Para ello, la herramienta deberá proporcionar un botón que al ser pulsado mostrará una ventana donde el usuario podrá editar los elementos que compondrán la anotación asociada al texto seleccionado: el título de la anotación y el texto de la anotación. Intercalado en el texto, se deberá permitir incluir contenido multimedia. Finalizada la edición, la herramienta deberá permitir visualizar la anotación al pulsar sobre el texto anotado.

3.2.10 Edición de un libro EPUB

Esta funcionalidad deberá permitir crear un libro digital de acuerdo al formato EPUB. Para ello, la herramienta proporcionará un botón que permita añadir secciones al libro. Las secciones creadas en un libro, se mostrarán en un listado ordenado (que será el mismo orden que se usará cuando se genere el libro en formato EPUB). Cuando se pulse sobre el botón de inserción, se mostrará un nuevo ítem en el listado de secciones que el usuario deberá rellenar con un nombre. A continuación, se deberá mostrará un área de la pantalla de edición que se corresponderá con la sección definida, y en la cual el usuario podrá añadir el contenido asociado a dicha sección. La herramienta deberá usar algún efecto visual para distinguir en la ventana de edición, las diferentes secciones definidas (por ejemplo, con un color de fondo diferente).

3.2.11 Mapa de navegación

Esta funcionalidad tendrá como objetivo mostrar un mapa de navegación de las páginas que conforman el libro digital que se está editando. En este sentido, el mapa permitirá representar como nodos las páginas web del libro y como aristas dirigidas los vínculos entre las distintas páginas web (Figura 3.1).

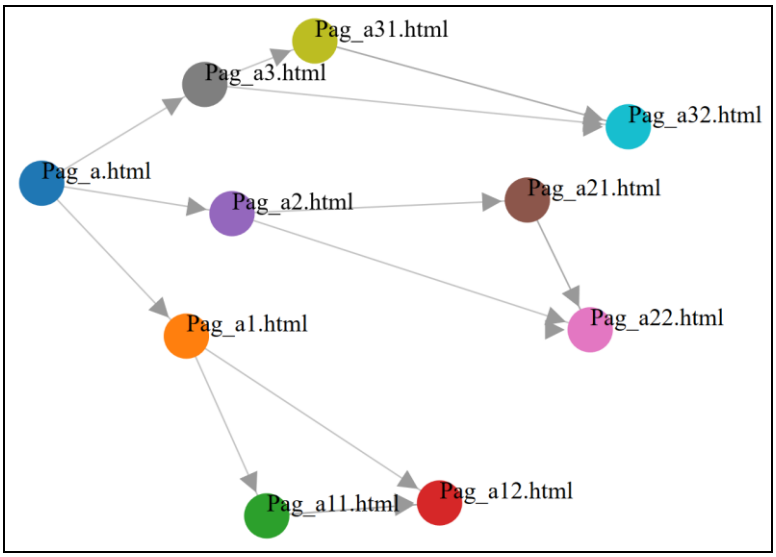


Figura 3.1 Ejemplo de mapa de navegación.

Para ejecutar esta funcionalidad, se le pedirá al usuario a través de una ventana, que seleccione la carpeta en la que se encuentran los contenidos del libro digital que se desean consultar.

3.2.12 Publicación de la obra

El contenido editado en la herramienta, se podrá guardar como una página web en formato HTML5 comprimida en un archivo Zip o bien en un archivo en formato EPUB.

En el primer caso, se dispondrá de una barra de herramientas que facilitará una opción de guardar como archivo comprimido. Al pulsar dicha opción, se abrirá una ventana para seleccionar la carpeta en la que se quiere guardar el archivo comprimido e introducir el nombre del mismo. La ventana dispondrá de un botón “Guardar” que al ser pulsado generará un archivo Zip que contendrá todos los documentos necesarios para el correcto funcionamiento de la página web:

- Una página HTML con el contenido del libro digital creado utilizando la herramienta.
- Una carpeta llamada *Audio* con todos los ficheros de audio insertados en el libro.
- Una carpeta llamada *Images* con todas las imágenes añadidas en el libro.
- Una carpeta llamada *stretchDocGen* con las siguientes carpetas:
  - js: con los ficheros de JavaScript jquery-3.1.1.min.js y stretch.text.js, necesarios para que el *stretchtext* funcione correctamente.
  - stretchDocGen.css: Hoja de estilos del libro generado.

En el segundo caso, se dispondrá de una barra de herramientas que facilitará una opción de guardar en formato EPUB. La herramienta deberá permitir guardar el contenido en este formato tanto si el contenido tiene definida una estructura en base a secciones como si no existe estructura alguna definida. En este último caso, la herramienta generará un único capítulo que contendrá a todo el texto editado. Con independencia de si el contenido tiene o no estructura, a continuación, se mostrará una ventana para seleccionar la carpeta en la que se quiere guardar el archivo e introducir el nombre del mismo. La ventana dispondrá de un botón "Guardar" que al ser pulsado generará el archivo en formato EPUB.

3.2.13 Previsualización de un libro

Esta funcionalidad facilitará al usuario la posibilidad de comprobar el funcionamiento y el resultado de la edición realizada con la herramienta. El proceso de previsualización será distinto en función de si el contenido se guarda como una página web o como un archivo EPUB.

Con independencia del caso, la herramienta dispondrá de un botón "Previsualizar". Si se trata de un libro en formato de página web, se lanzará una instancia del navegador que el usuario tenga configurado por defecto que mostrará una versión preliminar del libro. Y si se trata de un libro en formato EPUB, se lanzará una ventana para preguntar al usuario qué sección del libro se desea visualizar mediante una lista con todas las secciones que conforman el libro.

3.3 Arquitectura de la aplicación

Se trata de una aplicación de escritorio en la que se distinguen dos capas, la lógica de presentación y la lógica de negocio (Figura 3.2).

La lógica de presentación está formada por el editor web TinyMCE [TinyMCE] y las extensiones que se han realizado a su funcionalidad. Esta capa se encarga de atender a todos los eventos que lanza el usuario además de mostrar y actualizar la interfaz gráfica. Si los eventos incluyen algún tipo de procesamiento sobre el texto contenido en el área de texto de ILSAditor, entonces, en lugar de atender la llamada, invoca a la lógica de negocio.

La lógica de negocio está compuesta de todas las clases que hacen posible que el editor pueda ejecutar todas las funcionalidades solicitadas por el usuario del editor. Esta capa gestiona las funcionalidades de carga, publicación y previsualización de los documentos creados con el editor.

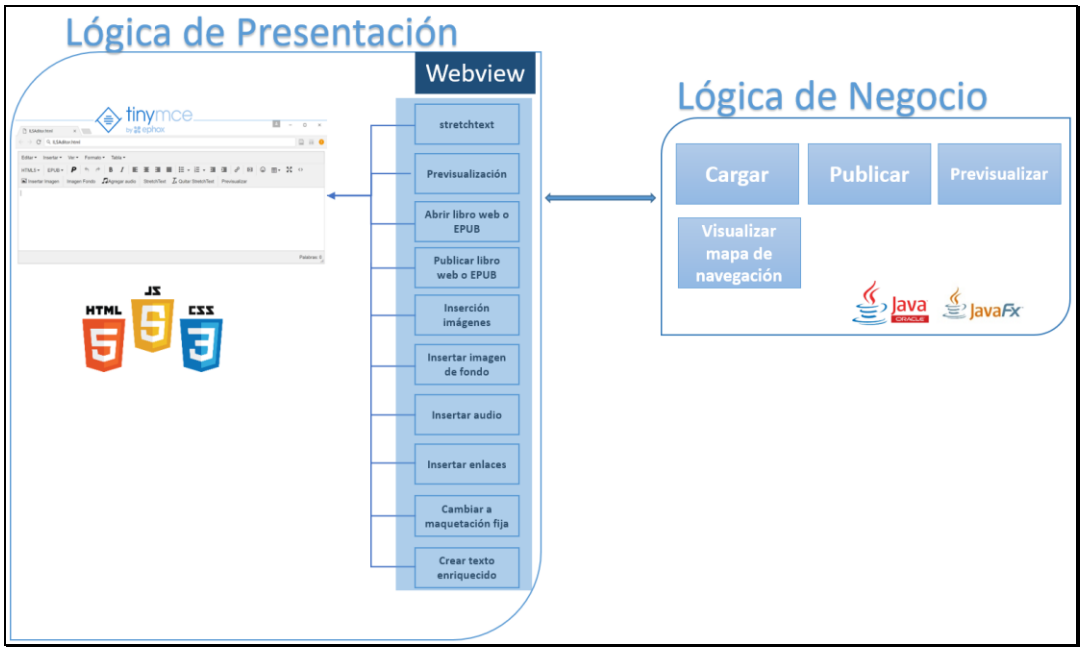


Figura 3.2 Arquitectura del Sistema

3.3.4 Tecnologías de la lógica de presentación.

Para la capa de presentación se han utilizado los lenguajes HTML, JavaScript y CSS, y se ha tomado como punto de partida el editor TinyMCE (Tiny Moxiecode Content Editor) [TinyMCE]. Se trata de un editor de texto de código abierto y distribuido gratuitamente bajo la licencia LGPL, aunque también tiene las versiones de pago "plugins premium" con un coste de 0.40 dólares por usuario activo al mes, "pro bundle" con un importe de 1 dólar (incluye además soporte del editor) o la versión "enterprise". Lo creó una compañía de software sueca fundada en el año 2003 llamada Moxiecode. La primera versión de TinyMCE se lanzó el 11 de marzo de 2004.

TinyMCE es un editor implementado íntegramente en JavaScript, HTML y CSS. Permite que la edición del texto se realice viendo directamente el resultado final (editor WYSIWYG). Es uno de los editores HTML más utilizados (prueba de ello es que se utiliza en Gestores de Contenidos Educativos como Moodle [Moodle] o sistemas de gestión de contenidos tan conocidos como Joomla [Joomla] o WordPress [WordPress] o en aplicaciones como Evernote [Evernote]).

3.3.5 Tecnologías de la lógica de negocio.

Para la lógica de negocio se ha utilizado el lenguaje Java en su versión 8. Además, para comunicar la lógica de presentación con la lógica de negocio, se ha utilizado el componente WebView de JavaFX.

JavaFX [JavaFX] es un framework de código abierto basado en Java que permite la construcción de aplicaciones de internet enriquecidas (RIAs, acrónimo de la expresión inglesa Rich Internet Applications) multiplataforma. Se considera el sucesor de Swing en el área de la construcción de interfaces gráficas de usuario usando la plataforma Java.

En particular, JavaFX proporciona un navegador embebido que a través de su API permite la visualización y navegación de las páginas HTML cargadas. Está basado en el motor de navegación de código abierto Webkit [WebKit]. El navegador embebido está formado por las clases que están dentro del paquete javafx.scene.web.package. El esquema de su arquitectura se puede observar en la Figura 3.3 [Oracle]

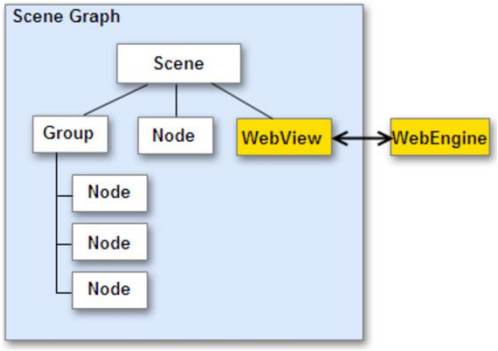


Figura 3.3 Arquitectura del navegador embebido: Scene Graph. Fuente: [Oracle docs]

Donde:

- Scene Graph: Es una estructura de datos en árbol. Muestra una colección de nodos dispuestos de forma jerárquica que representan elementos visuales.
- Scene: Es un contenedor de objetos visuales con los que el usuario puede interactuar. Se sitúa en la raiz de la jerarquía.
- Node: Cada elemento del grafo de escena es un nodo. Pueden ser controles de interfaz de usuario, contenido web, multimedia o gráficos. Si tiene hijos se llama nodo rama, por el contrario, si no tiene se llama nodo hoja
- Group: Es un contenedor especial que se encarga de agrupar un conjunto de nodos hijos.
- WebView: Gestiona la carga del contenido de la página HTML5, los eventos de entrada lanzados por el usuario, la aplicación de los estilos definidos en las hojas de estilo y la creación del DOM. La ejecución de estas tareas las realiza el WebEngine.
- WebEngine: Además de la realización de las tareas anteriores, realiza otras como mantener el historial de navegación de una sesión, o visualizar las ventanas emergentes.

[Sharan, 2015] añade un nivel más de abstracción al esquema de la arquitectura descrito anteriormente (Figura 3.4).

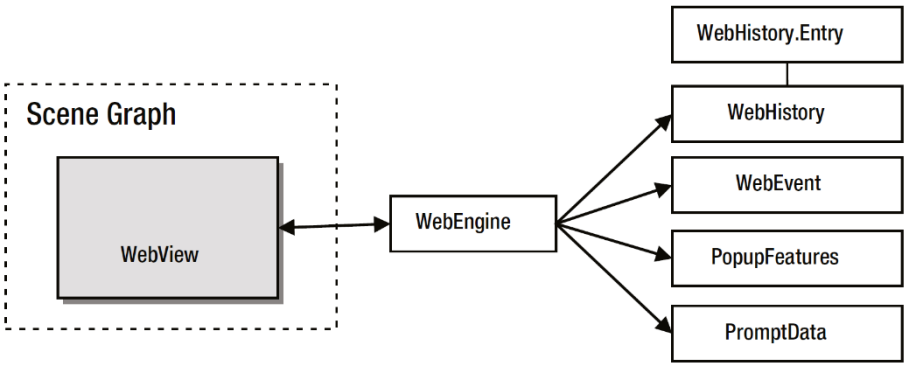


Figura 3.4 Arquitectura del navegador embebido.

El navegador web incluye una API con las siguientes clases en el paquete javafx.scene.web:

- WebHistory: Historial de las páginas visitadas en una sesión.
- WebHistory.Entry: Representa una entrada en el historial del navegador.
- WebEvent: Representa un evento generado por el WebEngine mientras procesa una página web.
- PopupFeatures: Encapsula los detalles de una ventana emergente.
- PromptData: Esta clase encapsula los datos pasados en el método prompt() del objeto window de JavaScript.

3.4 Implementación del ILSAditor

En esta sección se presentan cómo se ha llevado a cabo la implementación de la herramienta ILSAditor.

3.4.1 Funcionalidades como extensiones de TinyMCE

Las funcionalidades de edición de la herramienta se han implementado como una extensión de la funcionalidad propia de TinyMCE. Para desarrollarlo se han tenido que modificar o extender los siguientes elementos gestionados por el editor:

- Documento ILSAditor.html: Encargado de cargar el editor web TinyMCE y todas las extensiones de su funcionalidad que se han implementado a lo largo de este proyecto.

- Hojas de estilo CSS:
    - Las propias del editor TinyMCE para cambiar el aspecto de los botones y menús,
    - Las creadas por este proyecto para modificar la presentación de las etiquetas HTML contenidas dentro del área de texto y las de los libros generados con el editor. En el "Apéndice I - Manual de Usuario" de esta memoria en la sección I.5 Hojas de estilo se puede ver información más detallada relacionada con las hojas de estilo.
- Código JavaScript. Es el encargado de:
    - Atender a todos los eventos que lanza el usuario.
    - Invocar a la lógica de negocio, a través de la llamada a algún método de la clase java ServiceEditor que actúa como una fachada o distribuidor de las llamadas realizadas desde el frontend.
    - Actualizar los resultados (por ejemplo, inserta las etiquetas HTML necesarias para mostrar imágenes en la posición escogida por el usuario, reproducir audio o identificar una sección de un libro en formato EPUB).
    - Realizar comprobaciones sobre el texto interactivo.

3.4.2 Funcionalidades de la lógica de presentación implementadas

Las nuevas funciones implementadas en JavaScript son:

3.4.2.1 Abrir documento EPUB y HTML

Implementa la funcionalidad *Carga del libro digital*. Comprueba si el contenido del área de texto del editor está vacío. Si es así, invoca a la lógica de negocio para que abra un documento y se cargue el contenido del fichero en el editor. Si por el contrario contiene texto, lanza una ventana preguntando al usuario si: 1) desea abrir otro documento y guardar el contenido actual o 2) descartar el contenido (no guardar el texto) y abrir otro documento. Si escoge la primera opción invocará a la clase java serviceEditor del backend para que guarde el libro y posteriormente abra uno nuevo. Si elige la segunda, vacía el área de texto del editor y, posteriormente, llama a la clase serviceEditor para que cargue el libro

en formato epub o HTML. La implementación de esta funcionalidad se puede ver en las secciones 3.4.3.4 OpenHtml y 3.4.3.7 OpenEpub.

3.4.2.2 Guardar como HTML

Implementa la funcionalidad Generar libro en formato HTML. Comprueba si hay contenido en el área de texto del editor. Si lo hay, se llama a la clase handleSaveFile del backend para proceder a generar el documento HTML y guardarlo en un fichero comprimido .zip con todos los ficheros necesarios para su correcto funcionamiento y visualización. El detalle de la implementación de la parte del backend, se puede consultar en la sección 3.4.3.5 SaveHtml.

3.4.2.3 Guardar como EPUB

Implementa la funcionalidad *Generar libro en formato EPUB*. Comprueba si el contenido del área de texto del editor contiene el class "XHTMLfile". Si es así, quiere decir que el texto ya tiene el formato EPUB, por lo que invocará al método saveAsEpub() de la clase serviceEditor para crear el libro. Si no lo contiene, llamará al método checkIsEpub() de la clase serviceEditor para que informe al usuario de que si desea guardar el texto en formato EPUB tiene que escribir el nombre del capítulo. Si el usuario responde que sí, se creará una sección capítulo que contendrá todo el texto en el que únicamente tiene que editar el nombre del capítulo para poder generar el libro.

El método saveAsEpub() mencionado anteriormente invoca al método saveDoc() de la clase SaveEpub. El detalle de la implementación de esta clase se puede consultar en el apartado 3.4.3.8 Clase SaveEpub.

3.4.2.4 Añadir Metadata

Se comprueba si el primer hijo del elemento body contiene el atributo metadataInfo. Si lo contiene, se envía todo el nodo a la lógica de negocio (método openMetadata() de la clase serviceEditor, que a su vez llamará a la clase GuiMetadata). De lo contrario se envía vacío. A continuación, se sustituye el nodo que contiene el atributo metadataInfo por los nuevos valores que devuelve el método openMetadata().

La implementación de la clase GuiMetadata se puede ver en el apartado 3.4.3.9 GuiMetadata.

3.4.2.5 Insertar Imagen

Implementa la funcionalidad *Insertar imágenes en un nivel de stretchtext*. Desde el frontend se llama al método openImage() de la clase java ServiceEditor que invoca a su vez, a la clase GuiImage. El detalle de la implementación se puede consultar en el apartado 3.4.3.12 Clase GuiImage.

3.4.2.6 Insertar Audio

Implementa la funcionalidad *Insertar audio.* Se llama al método openAudio() de la clase ServiceEditor con el fin de abrir un cuadro de diálogo para permitir al usuario escoger el audio que se va a reproducir al pulsar en una imagen, botón o texto *stretchtext* previamente seleccionado en ILSAditor. Este método devuelve la ruta absoluta en la que se encuentra el fichero de audio seleccionado. Con esta ruta, se compone la etiqueta audio de HTML. La reproducción del audio se realiza capturando el evento “clic” del ítem pulsado.

3.4.2.7 Insertar Imagen de Fondo

Implementa la funcionalidad *Insertar imagen de fondo*. Se llama al método openBackgroundImage() de la clase ServiceEditor con el fin de abrir un cuadro de diálogo para permitir al usuario escoger la imagen que desea insertar como de fondo. Este método

devuelve la ruta absoluta en la que se encuentra la imagen que ha seleccionado. Con esta ruta, se compone la etiqueta img de HTML que se muestra en la Figura 3.5.

```
<img src=\"file:///" + img_tag + "\" class='HTMLtag' id='HTMLbackground' alt='imagen
de fondo' width='216' height='137'></img>
```

Figura 3.5 Composición de la etiqueta imagen de fondo.

Como se puede observar en la Figura 3.5, se añade el prefijo "file:///" y se reduce las dimensiones de la imagen para que la imagen se muestre correctamente y como una miniatura en el editor. Luego se inserta la etiqueta HTML img, en la primera posición del área de texto del editor para que la imagen se visualice en esa ubicación.

3.4.2.8 Insertar enlaces

Implementa la funcionalidad *Insertar enlace*. Se llama al método createLink() de la clase ServiceEditor con el fin de abrir una ventana emergente para permitir que el usuario configure las opciones relacionadas con el hiperenlace que desea insertar.

El detalle de la implementación de esta funcionalidad se puede consultar en 3.4.3.11 Clase GuiLink.

3.4.2.9 Stretchtext

En la implementación del *stretchtext* realizada en ILSAditor se llama stretch al texto que sugiere al lector el contenido que se mostrará si se clica sobre él y, se llama stretchable, al texto que se muestra u oculta, en función de si el usuario pulsa en el texto stretch. En la Figura 3.6, se muestra cómo se visualiza el *stretchtext* cuando se carga la página web o se oculta el texto stretchable. Así, el stretch se muestra resaltado en negrita junto a un botón de color verde y la parte stretchable oculta.

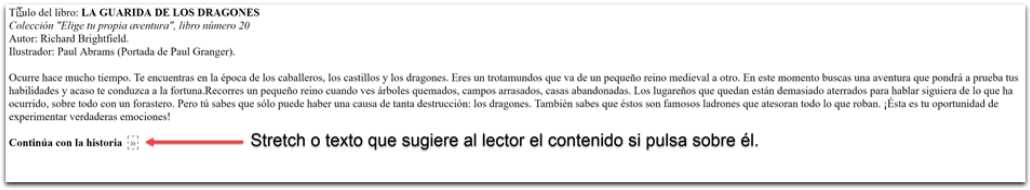


Figura 3.6 Ejemplo de *stretchtext* con texto stretchable oculto.

En la Figura 3.7, se muestra el resultado de pulsar sobre el stretch, es decir, el contenido del stretchable visible y el icono del stretch distinto para indicar que ya se ha expandido su contenido.

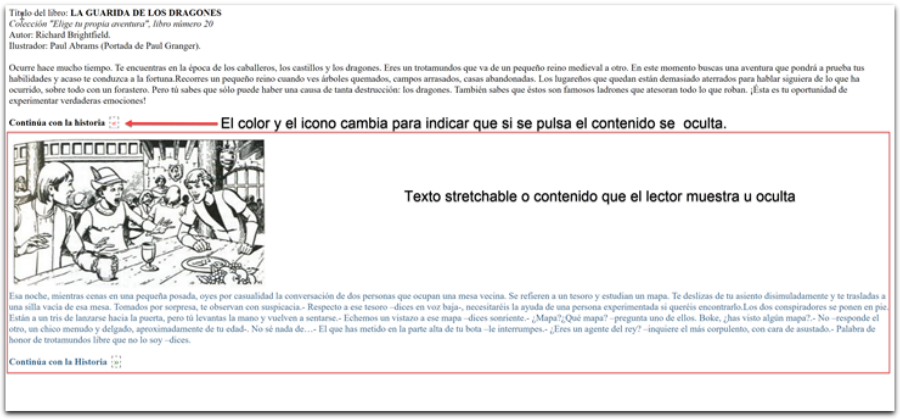


Figura 3.7 Ejemplo de *stretchtext* con texto stretchable develado.

Respecto a la codificación, cuando el usuario pulsa sobre el botón “Insertar *stretchtext*”, en primer lugar, se comprueba si la selección de texto que realiza el usuario está contenida en alguna capa. Si es así, se obtiene el número de la capa de profundización y se añade una unidad para calcular el nivel del *stretchtext* que se va a crear. Por el contrario, si no está contenido por ningún nivel es porque se trata del primer nivel. Esta comprobación inicial se realiza en la capa de presentación para aplicar el estilo correcto al nivel.

En segundo lugar, se envuelve el texto con la etiqueta *stretchtext* utilizando un elemento div de HTML, tal y como se muestra en la Figura 3.8.

```
<div class="stretch">
        <b>  Texto_Stretch   </b></div>
<div class="stretchable ' + nivel_de_stretchtext + '"> 'texto_seleccionado'</div>
```

Figura 3.8 Implementación del botón Insertar *stretchtext*.

Para que el contenido de la etiqueta HTML div no se muestre en la siguiente línea, en las dos hojas de estilo, se asigna a la propiedad “display” el valor “inline”, para modificar su comportamiento y forzar a que se muestre en la misma línea.

Por último, es necesario implementar la funcionalidad propia del *stretchtext*, es decir, que el texto se oculte o se devele en función de si el usuario pulsa o no en el texto (Figura 3.9).

```
$(document).ready(function(){
  $('.stretchable').hide();
    $('.stretch').click(function(evento){
      if (!$(this).hasClass('hidden')){
        $(this).parent().children(".stretchable").show();
      }
      else {
        $(this).parent().children(".stretchable").hide();
      }
      $(this).toggleClass("hidden");
      $(this).children(".stretch").toggleClass("hidden");
       });
});
```

Figura 3.9 Implementación de *stretchtext*.

3.4.2.10 Inserción de anotaciones

La opción del menú “Crear anotación” permite agregar ficción interactiva a la obra que se está editando. Las anotaciones son ventanas emergentes con contenido multimedia, utilizados para explicar o añadir información adicional a un elemento, llamado elemento anotado.

Una vez que se pulsa sobre el botón mencionado anteriormente, el editor revelará con un fondo de color gris el título y cuerpo de la anotación.

```
<div id="myModal1" class="modal fade">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <button class="close" type="button" data-dismiss="modal">&times;</button>
        <h4 class="modal-title">ESCRIBE EL T&Iacute;TULO AQU&Iacute;</h4>
      </div>
      <div class="modal-body">ESCRIBE EL TEXTO AQU&Iacute;</div>
    </div>
  </div>
</div>
```

Figura 3.10 Implementación de la anotación.

La Figura 3.10, muestra la implementación de esta funcionalidad implementada utilizando la biblioteca de Bootstrap. Las clases de Bootstrap utilizadas, se comentan a continuación:

- modal: se indica que se va mostrar una ventana emergente.
- fade: Añade efecto de desvanecimiento al abrir y cerrar la ventana.
- modal-content: el contenido de la ventana emergente.
- modal-header: Cabecera de la ventana modal
- button: Se indica que el título va a tener únicamente el botón de cerrar ventana.
- modal-title: Título de la ventana.
- modal-body: Contenido de la anotación.

Para que la anotación se visualice y funcione correctamente, en la publicación de la página web, se incluye en la generación del fichero comprimido el script bootstrap.min.js y la hoja de estilos bootstrap.min.css.

3.4.2.11 Maquetación de tipo fija

El objetivo de la funcionalidad *Cambiar el tipo de maquetación a fija* consiste en facilitar el diseño de cuentos o libros destinados a publicarse en tabletas.

Para conseguir este fin se reduce el tamaño del área de texto del editor modificando la hoja de estilo del editor para que el área de trabajo se adapte a las dimensiones de 768x1024 píxeles y también, se modifica la hoja del documento digital generado para el contenido del mismo esté enmarcado en el tamaño de pantalla de una Tablet y no se adapte de forma automática a ningún dispositivo con dimensiones diferentes.

3.4.2.12 Previsualización

Desde el frontend se llama al método runPage de la clase ServiceEditor. Éste a su vez invoca al método runPage() de la clase DigitalDoc. Los detalles de la implementación de esta funcionalidad se pueden consultar en el apartado 3.4.3.2 Clase DigitalDoc.

3.4.4.13 Visualización mapa de navegación

Desde la lógica de negocio se invoca al método showNavigationMap() de la clase ServiceEditor. Esta a su vez, redirige la llamada al método openNavMap() de la clase CreateJsonFile. Esta funcionalidad se puede consultar en el apartado 3.4.3.13 CreateJsonFile.

3.4.3 Clases de la lógica de negocio implementadas.

En la Figura 3.11 se presenta el diagrama de las clases desarrolladas. Tal como se puede observar en el mismo, el acceso a los métodos de la lógica de negocio se realiza a través del objeto serviceEditor que es el encargado de recopilar la información necesaria para invocar al método del backend que ejecute la funcionalidad solicitada desde la interfaz gráfica. En el diseño de la aplicación, se ha empleado el patrón de diseño Strategy , el cual es utilizado para "definir una familia de algoritmos, encapsularlos y hacerlos intercambiables permitiendo que el algoritmo varíe independientemente de los clientes que lo utilicen" [Gamma et al, 1994].

A continuación, se describen las diferentes clases e interfaces que aparecen en el diagrama.

3.4.3.1 Interfaces SaveDocBehaviour y OpenDocBehaviour

Como se ha comentado anteriormente, el patrón Strategy encapsula familias de algoritmos, en esta aplicación. Las familias son la funcionalidades o comportamientos "guardar documento" y "abrir documento". La ventaja de utilizar este diseño es que otros objetos pueden reutilizar los comportamientos ya que no están ocultos dentro de ninguna clase y se pueden añadir otros nuevos sin necesidad de modificar las interfaces (SaveDocBehaviour o OpenDocBehaviour) o las clases que las usan.

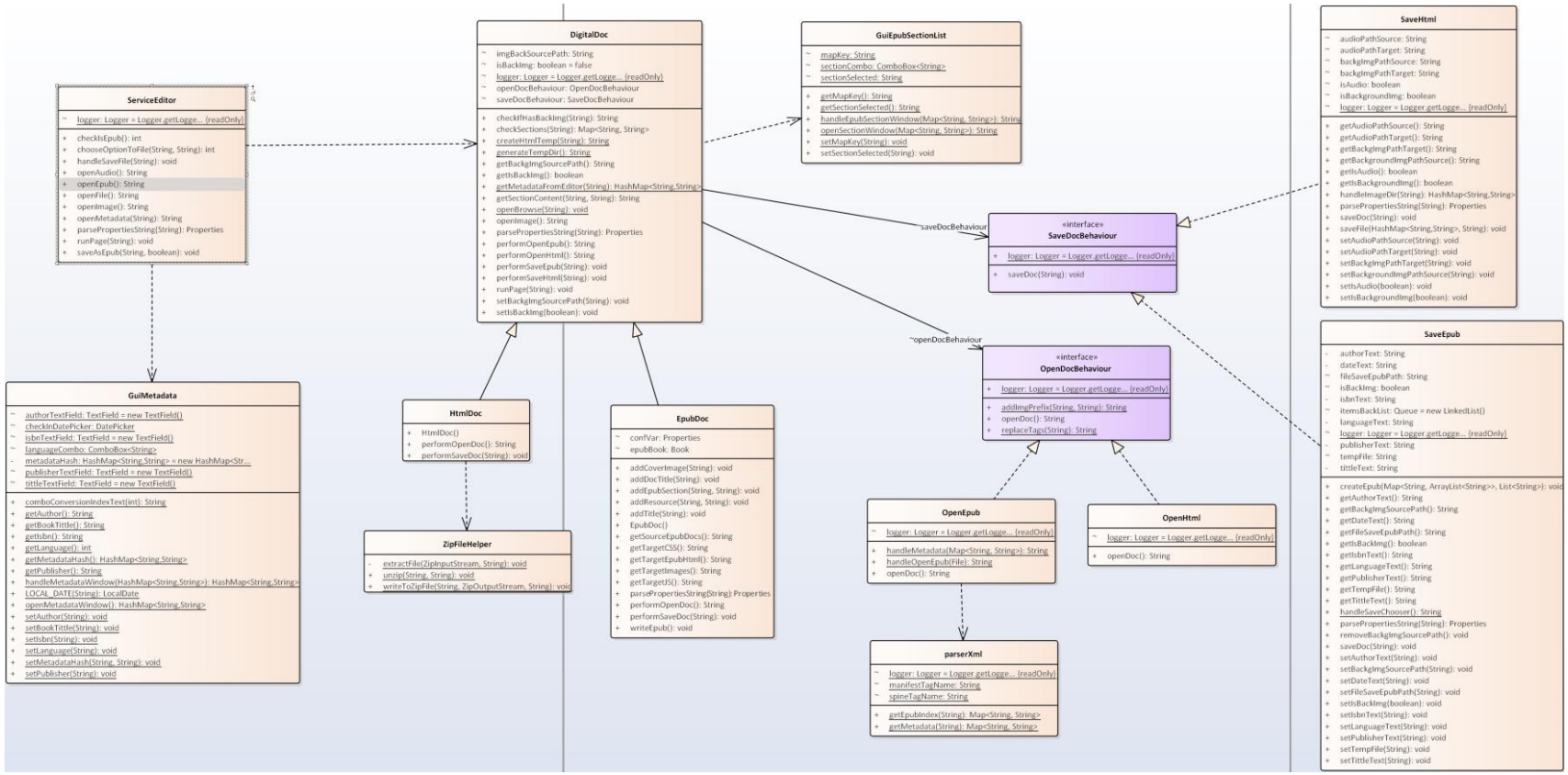


Figura 3.11 Diagrama de Clases de la lógica de negocio de ILSAditor.

3.4.3.2 Clase DigitalDoc

Contiene los métodos y propiedades que son comunes a todos los documentos digitales de la aplicación, es decir, los documentos HTML y los EPUB. El método más destacado de todos ellos es runPage() que implementa el caso de uso sobre la *Previsualización de un libro*. El método comienza comprobando cuantas secciones/capítulos contiene el libro. Si posee más de dos, se muestra una ventana para permitir que el usuario pueda elegir cuál de ellas quiere previsualizar (el detalle de esta implementación está en la clase GuiEpubSectionList). A continuación, genera un fichero HTML temporal. Comprueba el sistema operativo en el que se está ejecutando la aplicación y abre en una ventana del navegador la página HTML generada de forma temporal.

Finalmente, como se puede ver en el diagrama de clases, la clase *DigitalDoc* delega en las interfaces SaveDocBehaviour y OpenDocBehaviour, la implementación de las funcionalidades de guardar y abrir el documento a través de los métodos performSaveHtml, performSaveEpub, performOpenEpub y performOpenHtml respectivamente.

Las clases HtmlDoc y EpubDoc son implementaciones concretas de los documentos digitales, es decir, de esta clase que se está describiendo.

3.4.3.3 Clase HtmlDoc

Hereda las instancias SaveDocBehaviour y OpenDocBehaviour de la clase *DigitalDoc*. Usa la clase OpenHtml para gestionar la funcionalidad de abrir un documento HTML. Cuando se llama al método performOpenDoc se delega la responsabilidad de la apertura del documento al objeto OpenHtml. Por lo tanto, cuando se instancia HtmlDoc, su constructor hereda las variables de instancia OpenDocBehaviour en una nueva instancia de tipo OpenHtml, que es una implementación concreta de OpenDocBehaviour. El diseño para guardar el documento es el mismo que la apertura, pero con la interfaz SaveDocBehaviour en lugar de OpenDocBehaviour y SaveHtml en lugar de OpenHtml.

3.4.3.4 Clase OpenHtml

Implementa la interfaz OpenDocBehaviour. Es la encargada de implementar la funcionalidad *Carga del libro digital* en su método openDoc(). El método comienza abriendo un cuadro de diálogo para permitir al usuario escoger abrir un fichero con extensión txt, HTML o XHTML.

Si se trata de un archivo HTML realiza las siguientes modificaciones en el código:

- Se sustituye la etiqueta de salto de línea <br> por <br />, ya que es la etiqueta que puede interpretar el editor TinyMCE.
- Se comprueba el atributo src del contenido multimedia. Si la ruta del recurso es relativa se construye la ruta absoluta. Una vez que se tiene la ruta absoluta se añade el prefijo "file://" para que la imagen se visualice en el editor.
- Si se trata de una imagen de fondo, es decir, si en el elemento HTML <body> contiene el atributo background con el valor de la ruta en la que se encuentra la imagen, se convierte en una etiqueta img. A esta etiqueta se añade la ruta absoluta de la imagen en el atriburo src, el texto "imagen de fondo" en el atributo alt, el valor "HTMLbackground" en el atributo id y, finalmente, se reduce la dimensión de la imagen. Una vez realizadas todas las modificaciones se conseguirá que la imagen de fondo se visualice con un tamaño reducido (216x137) al principio del texto en el editor, se muestre el mensaje "imagen de fondo" cuando se posiciona el puntero del ratón sobre ella y se identifique esa imagen como de fondo para cuando genere/previsualice el libro la muestre correctamente.
- Se sustituye la etiqueta HTML de párrafo por la etiqueta . Se realiza esta sustitución porque si se intenta aplicar el *stretchtext* sobre la etiqueta p no funciona correctamente porque la etiqueta de bloque div no puede contener parte de la etiqueta p.

- Finalmente, el método termina devolviendo el código HTML adecuado o el texto plano al frontend para que lo muestre en el editor.

3.4.3.5 Clase SaveHtml

Implementa la interfaz SaveDocBehaviour. Es la encargada de implementar la funcionalidad Generar *el libro en formato HTML* en su método saveDoc(). El método comienza abriendo un cuadro de diálogo para permitir al usuario escoger la ubicación en la que se desea guardar el libro en formato comprimido Zip.

El proceso de generación del libro en formato web es el siguiente:

- Generar un HTML temporal teniendo en cuenta:
    - Modificar la ruta absoluta a relativa del atributo src de las imágenes y el audio.
    - Comprobar si se está ejecutando la aplicación en un sistema operativo Linux. Si es así, comprobar si el editor ha sustituido el $HOME de la ruta del atributo src por "../../../../../../../../". Si es así, reconstruir la ruta.
    - En el caso de que exista una imagen de fondo, hay que eliminar la etiqueta img y sus atributos, ya que pasa a utilizarse sólo la ruta relativa de la misma para añadirla como atributo (background) de la etiqueta HTML body.
    - El editor devuelve código HTML, pero sin las etiquetas HTML, head y body por lo que hay que añadir: la etiqueta HTML, head con su etiqueta meta, las referencias a la hoja de estilo utilizada en el libro (stretchDocGen/style/stretchDocGen.css), las etiquetas de los scripts necesarios para su correcto funcionamiento (stretchDocGen/js/jquery-3.1.1.min.js y stretchDocGen/js/stretch.text.js) y la etiqueta body.
    - Al final del texto obtenido del editor hay que añadirle las etiquetas HTML de cierre de body y HTML.
- Almacenar el HTML generado en una carpeta temporal. Esta creación se realiza de forma multiplataforma ya que la aplicación se puede ejecutar en cualquier sistema operativo.
- Comprimir los ficheros:

- En el caso de que exista imágenes crear en el fichero comprimido una carpeta (llamada “images”) y guardar una copia de las imágenes originales dentro de la carpeta.
- Si existe audio, crear una carpeta (llamada “audio”) y guardar una copia del audio original dentro de la carpeta.
- Crear una copia del HTML temporal y guardarlo en el raíz del fichero comprimido.
- Crear dentro del fichero comprimido la carpeta stretchDocGen. Dentro de ésta crear dos carpetas más, una llamada js para guardar los scripts y la segunda llamada style para guardar la hoja de estilos
- Crear una copia de los ficheros JavaScript (jquery-3.1.1.min.js y stretch.text.js) y guardarlos en la carpeta stretchDocGen\js.
- Crear una copia de la hoja de estilos (stretchDocGen.css) y guardarla en la carpeta stretchDocGen\style.

- Si los pasos anteriores se han generado correctamente, eliminar el fichero temporal.

3.4.3.6 Clase EpubDoc

Hereda las instancias SaveDocBehaviour y OpenDocBehaviour de la clase *DigitalDoc*. Usa la clase OpenEpub para gestionar la funcionalidad de abrir un documento HTML. Cuando se llama al método performOpenEpub se delega la responsabilidad de la apertura del documento al objeto OpenEpub. Por lo tanto, cuando se instancia EpubDoc, su constructor hereda la variable de instancia OpenDocBehaviour en una nueva instancia de tipo OpenEpub, que es una implementación concreta de OpenDocBehaviour. El diseño para guardar el documento es el mismo que la apertura, pero con la interfaz SaveDocBehaviour en lugar de OpenDocBehaviour y SaveEpub en lugar de OpenEpub.

3.4.3.7 Clase OpenEpub

Implementa la interfaz OpenDocBehaviour. Es la encargada de implementar la funcionalidad *Carga del libro digital* en su método openDoc(). El método comienza abriendo un cuadro de diálogo para permitir al usuario escoger abrir un fichero con extensión EPUB.

El proceso de apertura del libro es el siguiente:

- Cambiar la extensión del fichero EPUB a Zip.
- Descomprimir el fichero en un archivo temporal. Se crea una carpeta en el directorio de la aplicación con el nombre del fichero y posteriormente se descomprime todos los documentos dentro de esta carpeta.
- Buscar dentro de la carpeta el fichero con extensión opf. Se trata de un documento XML que describe la estructura del libro y contiene un elemento llamado spine, que contiene uno o más elementos itemref. Cada uno de estos elementos itemref hace referencia a un documento de contenido OPS designado en el manifiesto. El orden en que se muestra los itemref es el orden de lectura que se mostrará al lector del libro EPUB. Por ello, el siguiente paso es obtener todos los valores de los elementos itemref del elemento spine, para obtener el orden de visualización/lectura en el editor.
- El elemento manifest provee una lista de todos los ficheros que forman parte del libro. Por ello, se obtiene por el orden del spine, la ruta y nombre de los ficheros xhtml, que van a ser cada una de las secciones que componen el libro en el editor.
- Se obtienen todos los elementos que contiene el elemento metadata.
- Lo metadatos obtenidos en el apartado anterior se transforman a etiquetas HTML para mostrarlo posteriormente en el editor. La estructura es .
- A continuación, se abre cada uno de los ficheros que componen la publicación. Se eliminan las etiquetas HTML head, HTML y body, o lo que es lo mismo, se obtiene el contenido del body. Seguidamente, se envuelve el contendido de cada

uno de los documentos con la etiqueta para identificar todos los ficheros (XHTMLfile) y el nombre de cada uno de ellos por el id.

- Se realiza las siguientes modificaciones sobre el código HTML:
    - Se comprueba el atributo src del contenido multimedia. Si la ruta del recurso es relativa se construye la ruta absoluta. Una vez que se tiene la ruta absoluta se añade el prefijo "file://" para que la imagen se visualice en el editor.
    - Si se trata de una imagen de fondo, es decir, si en el elemento HTML <body> contiene el atributo background con el valor de la ruta en la que se encuentra la imagen, se convierte en una etiqueta img. A esta etiqueta se añade la ruta absoluta de la imagen en el atributo src, el texto "imagen de fondo" en el atributo alt, el valor "HTMLbackground" en el atributo id y, finalmente, se reduce la dimensión de la imagen. Una vez realizadas todas las modificaciones se conseguirá que la imagen de fondo se visualice con un tamaño reducido (216x137) al principio del texto en el editor, se muestre el mensaje "imagen de fondo" cuando se posiciona el puntero del ratón sobre ella y se identifique esa imagen como de fondo para cuando genere/previsualice el libro la muestre correctamente.
- Una vez que se abren y etiquetan todos los documentos xhtml se tiene todo el texto que forma el libro y ya se puede enviar al frontend para visualizarlo en el editor.

3.4.3.8 Clase SaveEpub

Implementa la interfaz SaveDocBehaviour. Es la encargada de implementar la funcionalidad *Generar libro en formato EPUB* en su método saveDoc(). El método comienza abriendo un cuadro de diálogo para permitir al usuario escoger la ubicación en la que se desea guardar el libro en formato EPUB.

El proceso de generación del libro es el siguiente:

Se obtienen todas las etiquetas relacionadas con los metadatos del libro, es decir, todos los hijos de la etiqueta HTML . Se obtiene el valor de cada una de ellas y se añaden en el documento content.opf como hijos del elemento metadata.

Por cada sección del libro o contenido de la etiqueta se genera un fichero XHTML en un directorio temporal que sea multiplataforma. En la creación de los ficheros se siguen los siguientes pasos:

- El nombre del fichero es el id de la etiqueta div mencionada en el apartado 2.
- El primer hijo de la etiqueta HTML siempre va a ser una etiqueta HTML h1, que recogerá el título de la sección. Este título aparecerá en la navegación de las tablas de contenido.
- Se modifica la ruta del atriburo src de los contenidos multimedia (imágenes y audio) de absoluta a relativa.
- En el caso de que exista una imagen de fondo, hay que eliminar la etiqueta img y sus atributos ya que pasa a utilizarse sólo la ruta relativa de la misma para añadirla como atributo (background) de la etiqueta HTML body.
- Se añade al principio del código HTML las etiquetas HTML: las etiquetas HTML, head con su etiqueta meta, las referencias a la hoja de estilo utilizada en el libro (stretchDocGen/style/stretchDocGen.css), las etiquetas de los scripts necesarios para su correcto funcionamiento (stretchDocGen/js/jquery-3.1.1.min.js y stretchDocGen/js/stretch.text.js) y la etiqueta body.

Una vez creados todos los ficheros XHTML, se procede a crear el fichero comprimido con extensión EPUB:

- Se añaden el audio (en el directorio text/audio) y las imágenes (en el directorio text/images) si existen.
- Se añaden los ficheros js: text/js/jquery-3.1.1.min.js, text/js/stretch.text.js y la hoja de estilos (text/style/stretchDocGen.css)
- Se añaden todos los ficheros XHTML.

3.4.3.9 Clase GuiMetadata

El objetivo de esta clase es mostrar al usuario una ventana para permitirle crear, modificar o leer los metadatos del libro.

Antes de cargar la ventana comprueba si el libro ya contiene algún metadato. En caso positivo, carga en la ventana los metadatos existentes.

La ventana posee dos botones:

- Cancelar: No realiza ninguna modificación en el valor de los metadatos.
- Guardar: Modifica el valor de los metadatos originales por el valor que ha escrito el usuario.

Los valores de los metadatos se envían en formato HTML al frontend (en etiquetas HTML como hijos de la etiqueta) para mostrarse ocultos al principio del texto.

El objetivo es que si se genera un libro EPUB se puedan capturar los metadatos cuando se procesa el texto y, si se genera un libro HTML, dar al usuario de la aplicación la opción de tener los metadatos y presentarlos como desee modificando la etiqueta #metadataInfo del fichero stretch.css.

3.4.3.10 Clase GuiEpubSectionList

El objetivo de esta clase es mostrar al usuario una ventana para permitirle seleccionar la sección/capítulo que desea previsualizar.

Se comienza recorriendo el código HTML del libro para recuperar las secciones/capítulos que lo componen. Se muestra al usuario una ventana con una lista de capítulos para que seleccione uno de ellos a previsualizar.

Una vez que haya seleccionado el ítem, se envía el texto al método runPage de la clase DigitalDoc.

3.4.3.11 Clase GuiLink

El objetivo de esta clase es mostrar al usuario una ventana para permitirle crear un enlace sobre un elemento seleccionado en el editor.

En la ventana se muestra las siguientes opciones:

- Introducir manualmente una URL. Con los datos introducidos por el usuario se construye la etiqueta <a> de HTML.
- Seleccionar un fichero HTML del sistema de ficheros local con el que se quiere vincular el recurso. Se trata de un checkbox, si se pulsa sobre él, se abrirá una ventana para permitir escoger un documento (HTML) del directorio de trabajo.

  Una vez seleccionado el documento, se recoge la ruta relativa del mismo para construir la etiqueta <a> de HTML.
- Crear un documento HTML vacío para vincularlo con el texto seleccionado el editor. Es un checkbox, si se pulsa sobre él, se abrirá una ventana para escoger el directorio en el que se quiere crear un documento HTML vacío. Una vez seleccionado el directorio y escrito el nombre de la nueva página web, se realizan los siguientes pasos:
  - Se crea un fichero con extensión HTML.
  - Se escribe y guarda el fichero con las etiquetas básicas de HTML (HTML, head, body).
  - Se obtiene el path relativo del nuevo fichero creado y se construye la etiqueta <a> de HTML.

Las etiquetas <a> de HTML construidas en los puntos anteriores es la cadena que se retorna al método de la lógica de presentación llamante.

3.4.3.12 Clase GuiImage

El objetivo de esta clase es mostrar al usuario una ventana emergente para permitirle insertar y editar las propiedades de una imagen. Las opciones de personalización de la imagen que se muestra al usuario son las siguientes:

- Introduce la dirección de la imagen:
    - Introducir manualmente la URL de la imagen. Con esta opción se construye el atributo src de la etiqueta img.
    - Seleccionar una imagen del sistema de ficheros local a través del botón "Buscar". Con esta opción se obtiene la ruta relativa de la imagen para componer el atributo src de la etiqueta img.
- Descripción emergente (tooltip): Con esta opción se construye el atributo tittle de la imagen.
- Espacio vertical y horizontal: Con esta opción se construye la propiedad margin de CSS para crear el margen alrededor de cada imagen. Con los valores de las propiedades del margen vertical y horizontal se crea las propiedades de cada lado de la imagen.

  Con el margen vertical se crean las propiedades: margin-top y margin-bottom. Con el margen horizontal se crean: margin-right y margin-left.
- Posición de la imagen: Se refiere a la posición de la imagen en relación al texto. Con esta opción se asigna un valor a la propiedad CSS float. Los valores pueden ser los siguientes: none (La imagen se muestra aislada del texto), right (El texto se sitúa a la izquierda de la imagen) o left (El texto se sitúa a la derecha de la imagen).
- Enlazar con la URL: Añade un enlace sobre la imagen con la que se está trabajando. Permite las opciones comentadas en la sección 3.4.3.11 Clase GuiLink.

Con toda la información recogida en los puntos anteriores construye las etiquetas img y a de HTML, con las propiedades que haya escogido el usuario. Esta información es la que se retorna al método llamante del frontend.

3.4.3.13 Clase CreateJsonFile

El objetivo de esta clase es la de mostrar el mapa de navegación de las páginas web que componen una obra. El proceso para la generación del mapa es el siguiente:

- Mostrar una ventana para permitir al usuario seleccionar el directorio que contiene los documentos HTML y obtener su ruta.
- Se construye un documento JSON con la misma estructura contemplada en el ejemplo de la Figura 3.12, con las páginas web del directorio seleccionado.


```
{
  "links": [
    {
      "source": 0,
      "target": 1
    },
    {
      "source": 0,
      "target": 2
    },
    {
      "source": 1,
      "target": 2
    },
    {
      "source": 2,
      "target": 3
    },
  ],
  "nodes": [
    {
      "name": "Pag_1.html",
      "id": 0
    },
    {
      "name": "Pag_2.html",
      "id": 1
    },
    {
      "name": "Pag_3.html",
      "id": 2
    }
  ]
}
```


Figura 3.12 Ejemplo de documento JSON.

La estructura del documento JSON, está compuesto por el arrays “links” y “nodes”. “links” contiene el identificador (id) del documento que contiene el enlace (llamado “source”) y el id del documento vinculado (“target”). El array “nodes” contiene el nombre de la página web (“names”) y su id (“id”).

- La visualización del mapa de navegación se realiza abriendo el documento NavigationMap.html. Esta página web contiene el código fuente necesario para

obtener los datos almacenados en el documento JSON generado anteriormente, y pintar los nodos (páginas HTML) y sus relaciones (aristas dirigidas) mediante la utilización de la biblioteca gráfica d3js. La implementación del código de esta página web se ha basado en las explicaciones de [Fancellu, 2017].

3.4.4 Comunicación entre el backend y el frontend.

Para lograr esta comunicación entre la lógica de negocio y la lógica de presentación usando WebView, se han seguido las indicaciones de [Dea C. et al, 2014] que aparecen descritas en el código de ejemplo de la Figura 3.13.

```
JSObject jsWindow = (JSObject)webView.getEngine().executeScript("window");
clase_java string_clase_Java = new clase_java ();
jsWindow.setMember("clase_java", string_clase_Java);
```

Figura 3.13: Código de ejemplo para establecer una conexión.

Como se puede observar en la Figura 3.13, en primer lugar, se obtiene el objeto WebEngine, que en este caso es JSObject, a través de la llamada al método executeScript("window"). El JSObject es un objeto encapsulado que actúa como puente entre JavaScript y Java 8.

Una vez que se ha obtenido el JSObject, se puede pasar cualquier objeto al contexto del motor de JavaScript a través de la invocación del método setMember() pasándole un string y el objeto que se quiere invocar.

Una vez lograda la conexión, se procede a cargar y mostrar en el WebView la página web (Figura 3.14).

```
webView.getEngine().load(URI_del_fichero);
```

Figura 3.14 Ejemplo de implementación de carga y visualización de una página web utilizando WebView.

Una vez implementada esta última línea de código, la conexión entre ambas tecnologías ya está establecida.

3.5 Conclusiones

En este capítulo se ha descrito la especificación de requisitos de la herramienta desarrollada en el presente trabajo fin de master, basada en el análisis de los problemas y necesidades presentes en la edición digital comentados en el capítulo anterior.

Se ha comentado que la arquitectura de la aplicación se estructura en capas para mantener desacopladas las partes que componen la herramienta. La primera capa es la lógica de presentación formada por el editor web TinyMCE [TinyMCE] y las extensiones que se han realizado a su funcionalidad. La segunda capa, es la lógica de negocio compuesta de todas las clases que hacen posible que el editor pueda ejecutar las funcionalidades solicitadas por el usuario del editor.

También se ha explicado que la comunicación entre la lógica de presentación y la lógica de negocio se realiza con el componente WebView de JavaFX.

Finalmente, se ha ilustrado cómo se ha realizado la implementación de cada una de las necesidades descritas al comienzo de este capítulo.

Capítulo 4 - Evaluación de la herramienta.

En este capítulo se muestran y analizan los resultados obtenidos en la evaluación inicial de la herramienta ILSAditor.

4.1 Descripción del experimento

Para realizar la evaluación inicial de la herramienta se organizó un taller de una hora y media de duración en la Facultad de Informática de la UCM a finales de junio de 2017. El taller se estructuró en dos partes. En la primera, se realizó una exposición de media hora en la que se explicó la funcionalidad y características del editor, y en la segunda parte se propusieron a los asistentes un conjunto de ejercicios prácticos orientados a poner en práctica las funcionalidades del editor, que previamente habían sido explicadas. Los ejercicios consistieron en la creación de un libro digital, la generación de varios niveles de *stretchtext* con texto e imágenes y la publicación de un libro en los formatos EPUB y web.

Finalizado el taller, los participantes rellenaron un cuestionario para evaluar la usabilidad de la herramienta, que en la siguiente sección se describe con detalle.

El número de participantes en el taller, fueron siete miembros del área de Humanidades de la Universidad Complutense de Madrid y de la Universidad Nacional de Educación a Distancia (UNED), donde cinco participantes eran profesores y dos eran estudiantes.

4.2 Instrumento de evaluación

El instrumento de evaluación utilizado en esta experiencia fue un cuestionario con diecinueve preguntas donde las respuestas posibles son cinco categorías correspondientes a una escala Likert [Likert, 1932] y una última pregunta donde la respuesta se deja abierta al criterio del participante.

La escala Likert utilizada en el cuestionario constaba de cinco valores de medición:

- 1: Ninguna o nada,
- 2: Alguna,

- 3: Normal,
- 4: Bastante
- 5: Mucha.

El objetivo del cuestionario era recoger información acerca de la percepción de los encuestados con respecto a cinco ejes de medición que se consideraron relevantes para evaluar la herramienta:

- Experiencia en el trabajo y las técnicas de edición digital.
- Usabilidad.
- Utilidad en términos generales de la herramienta.
- Utilidad de *stretchtext*.
- Mejoras de la herramienta.

Las preguntas utilizadas en el cuestionario se muestran en la Figura 6.1.

Preguntas
1. ¿Tienes experiencia en edición digital?
2. ¿Qué grado de experiencia tenías en las técnicas explicadas?
3. ¿Te ha parecido fácil usar la herramienta?
4. ¿Has necesitado las explicaciones de la ponente para el uso de la herramienta?
5. ¿Has necesitado usar el manual que se ha proporcionado para el uso de la herramienta?
6. ¿Has necesitado la ayuda de tus compañeros?
7. ¿Te ha parecido útil usar la herramienta?
8. ¿Te ha parecido claro el diseño y los menús de opciones?
9. ¿Preferirías tener más información a la vista?
10. ¿Te habías planteado antes de esta experiencia realizar strechtext en la edición digital?
11. ¿Te parecía importante utilizar strechtext en la edición digital antes de esta experiencia?
12. ¿Crees que el uso de strechtext ayuda a incentivar la lectura digital de un libro?
13. ¿Crees que el uso de esta herramienta para insertar strechtext ayuda a incentivar la lectura digital de un libro?
14. ¿Crees que es suficiente que la herramienta admita únicamente html y epub?
15. ¿Crees que es suficiente que la herramienta se ejecute de manera local y no admita el trabajo colaborativo?
16. ¿Crees que la herramienta te ha ayudado a realizar la actividad que se te ha propuesto en el taller y que te habría sido más difícil sin ella?
17. ¿Te resulta interesante y útil realizar esta actividad con tus compañeros?
18. ¿Ves interesante que la herramienta incluyera un editor de anotaciones críticas?
19. ¿Recomendarías esta herramienta?

Figura 6.1 Cuestionario de evaluación de ILSAditor.

4.3 Resultados obtenidos

En este apartado se presentan los resultados de un análisis descriptivo sobre la frecuencia de respuesta para las distintas categorías de cada una de las preguntas del cuestionario aplicado.

4.3.1 Resultados relacionados con la experiencia en edición digital

El grado de experiencia previa de los participantes en la edición digital y las técnicas explicadas en el taller, se evaluó a través de las preguntas uno y dos del cuestionario. Los resultados obtenidos se muestran en las figuras 6.2.a y 6.2.b.

a) b)

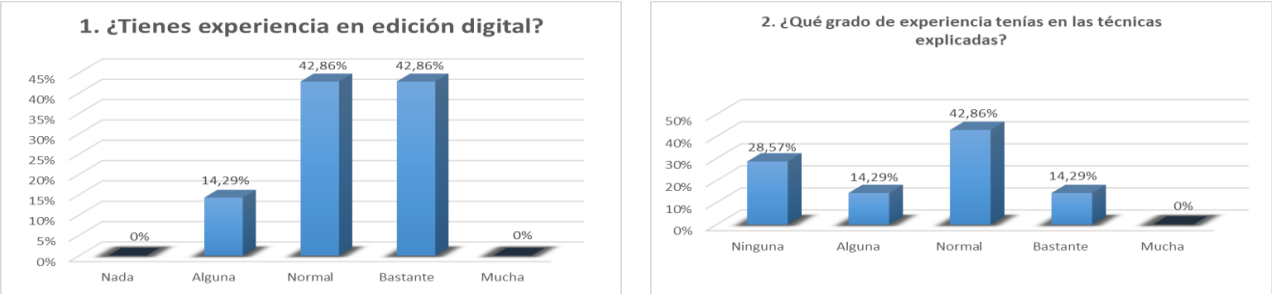


Figura 6.2. a) Respuestas a la pregunta 1. b) Respuestas a la pregunta 2.

En la Figura 6.2.a, se aprecia que la mayoría de los sujetos contaban con experiencia previa en edición digital tanto a un nivel medio (42.86% contestaron “Normal”) como un nivel más alto (42.86% “Bastante”).

Y en la Figura 6.2.b, se observa que la mayoría de los participantes tenían algún grado de experiencia en las técnicas explicadas, el 71,43%, mientras el 28.57% restante no tenía experiencia en estas técnicas.

4.3.2 Resultados relacionados con la usabilidad de ILSAditor

La usabilidad de la herramienta se evaluó mediante las preguntas 3, 4, 5, 6, 8 y 9 del cuestionario. Las preguntas se centran en aspectos tales como la facilidad de uso de la herramienta, y la necesidad de explicaciones o información adicional para desarrollar los ejercicios propuestos. Los resultados obtenidos para las preguntas 3, 4, 5, 6, 7 y 8 se muestran en las figuras 6.3.a, 6.3.b, 6.4.a, 6.4.b, 6.5.a y 6.5.b respectivamente.

a) b)

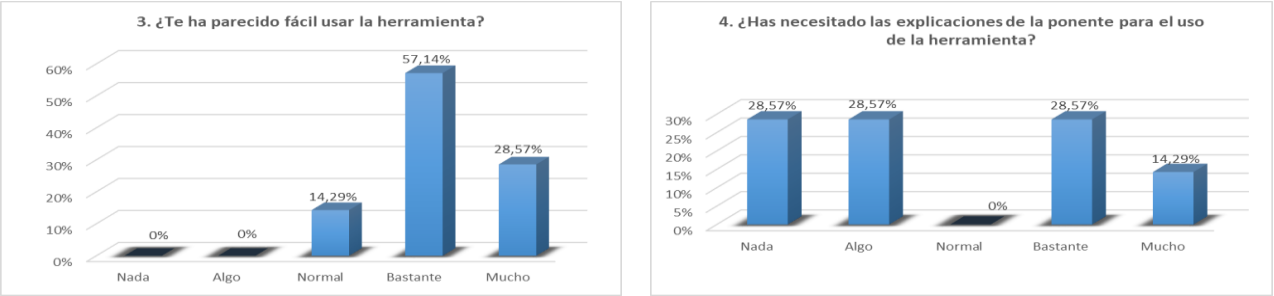


Figura 6.3. a) Respuestas a la pregunta 3. b) Respuestas de la pregunta 4.

En la Figura 6.3.a, se muestra que a la mayoría de los sujetos les ha parecido bastante fácil o muy fácil usar ILSAditor (85,7%), mientras que el resto manifiestan que el uso de la herramienta es “Normal”.

En la Figura 6.3.b, se aprecia que la mayoría de los participantes necesitaron pocas o ninguna explicación de la ponente para ejecutar sus ejercicios (57,13%). El resto de sujetos han necesitado bastantes explicaciones (28,7%) o muchas (14,29%).

a) b)

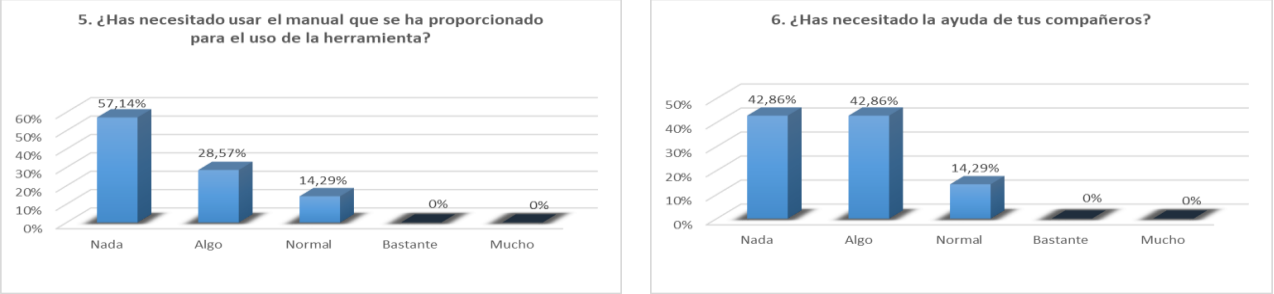
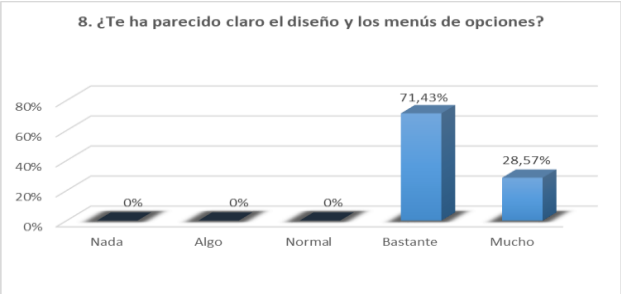


Figura 6.4. a) Respuestas a la pregunta 5. b) Respuestas a la pregunta 6.

En la Figura 6.4.a, se observa que la mayoría de los sujetos no ha necesitado usar el manual para realizar los ejercicios propuestos (57,14%) mientras que al resto les ha parecido que el uso de la herramienta es el “Normal” para este estilo de aplicaciones.

En la Figura 6.4.b, se aprecia que la mayoría de los participantes ha necesitado poca o ninguna ayuda de sus compañeros (85,71%), mientras que el resto han necesitado algo de ayuda adicional (14,29%).

a) b)



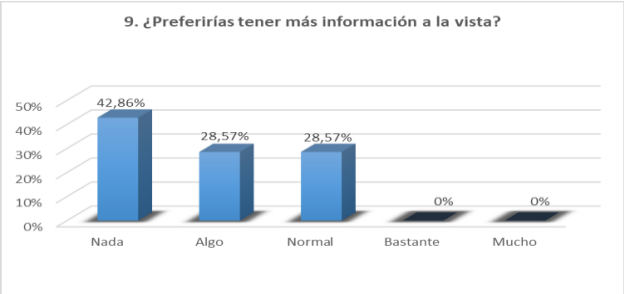


Figura 6.5. a) Respuestas a la pregunta 8. b) respuestas a la pregunta 9.

En la Figura 6.5.a, se observa que a todos los sujetos les ha parecido el diseño de la herramienta basteante claro o muy claro, 71,43% y 28,57% respectivamente.

En la Figura 6.5.b, se aprecia que el 57,13% de los participantes preferirían tener más información a la vista, como por ejemplo, una opción de ayuda añadida en el editor. Sin embargo, el resto no lo ve necesario.

Teniendo en cuenta que la escala de medición de las preguntas 4, 5, 6 y 9 se definen en sentido inverso al de las preguntas 3 y 8, entonces se han generado las gráficas 6.6.a y 6.6.b que representan el promedio de las puntuaciones obtenidas en cada una de las categorías.

a) b)

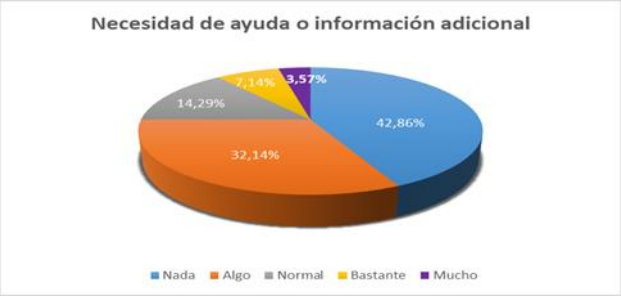


Figura 6.6. a) Respuestas a las preguntas 4, 5, 6 y 9. b) Respuestas a las preguntas 3 y 8.

En la Figura 6.6.a, se muestra que la mayoría de los participantes (92,85%) considera que la herramienta es muy fácil de usar y tiene un diseño muy claro.

Por último,en la Figura 6.6.b, se muestra que la mayoría de los sujetos (75%) considera que no ha necesitado información o ha necesitado poca para entender el funcionamiento de la herramienta.

4.3.3 Resultados relacionados con la utilidad de stretchtext

La evaluación de la funcionalidad del *stretchtext* de la herramienta se ha realizado mediante las preguntas 10, 11, 12 y 13 del cuestionario. Sobre este aspecto, se quería conocer la opinión de los participantes acerca de la importancia de utilizar la técnica del *stretchtext* antes y después de realizar el taller. Los resultados obtenidos para las preguntas 10, 11, 12 y 13 se muestran en las figuras 6.7.a, 6.7.b, 6.8.a, y 6.8.b respectivamente.

a) b)

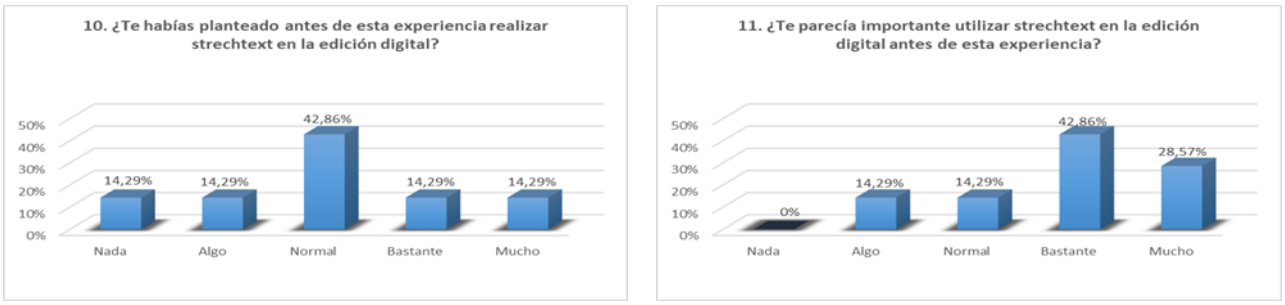


Figura 6.7. a) Respuestas a la pregunta 10. b) Respuestas a la pregunta 11.

En la Figura 6.7.a se aprecia que las frecuencias obtenidas en la pregunta 10 se distribuyen simétricamente, concentrando a la mayoría de los participantes en la categoría central ("Normal"), y el resto de sujetos distribuyéndose tanto a la izquierda como a la derecha de ésta, en igual proporción (14,29%).

En la Figura 6.7.b se observa que la mayoría de los participantes consideran importante ("Bastante") o muy importante ("Mucho") la utilización de *stretchtext* en la edición digital (71,42%).

En esta pregunta resulta llamativo que todos los participantes consideraban que era importante la utilización de *stretchtext* antes de conocer la herramienta ILSAditor (para la

edición digital). Sin embargo en la pregunta anterior no se planteaban la utilización de esta técnica en la edición digital.

a) b)

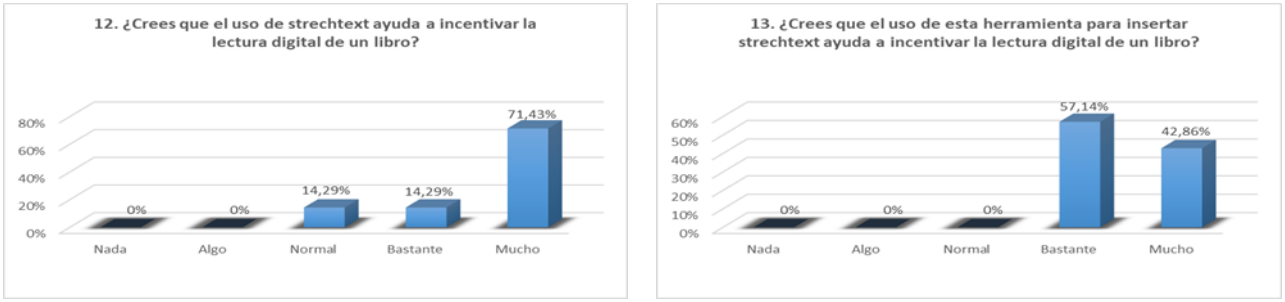


Figura 6.8. a) Respuestas a la pregunta 12. b) Respuestas a la pregunta 13.

En la Figura 6.8.a se ve que la mayoría de sujetos (71,43%) creen que el *stretchtext* ayuda "Mucho" a incentivar la lectura de un libro digital. El resto piensan que ayuda "Bastante" o "Normal", en la misma proporción.

En la Figura 6.8.b se aprecia que el 100% de los participantes creen que ILSAditor puede ayudar "Bastante" o "Mucho" a incentivar la lectura de un libro digital.

Teniendo en cuenta que la escala de medición de las preguntas 10 y 11 miden la percepción de la importancia del *stretchtext* antes de la realización del taller, y las preguntas 12 y 13, miden la utilidad del *stretchtext*, entonces se han generado las gráficas 6.9.a y 6.9.b que representan el promedio de las puntuaciones obtenidas en cada una de las categorías.

a) b)

Figura 6.9. a) Respuestas a la pregunta 10 y 11. b) Respuestas a la pregunta 12 y 13.

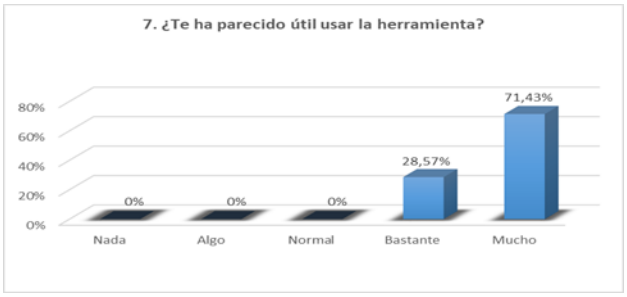
En la Figura 6.9.a, se observa que el 21,42% de los participantes no se había planteado o se lo había planteado muy poco la utilización del *stretchtext* mientras que el 78,58% restante, se lo había planteado "Bastante" (28,57), "Normal" (28,57%) y "Mucho" (21,43%).

Y por último, en la Figura 6.9.b se aprecia que la mayoría de los sujetos (92,84%) considera muy o bastante útil la utilidad del *stretchtext*.

4.3.4 Resultados relacionados con la utilidad general de la herramienta.

La utilidad general de la herramienta se ha evaluado mediante las preguntas 7, 16 y 19 del cuestionario. Sobre este aspecto, se quería conocer la valoración de los participantes acerca de si les había resultado útil usar la herramienta y les había ayudado a realizar las actividades propuestas. Los resultados obtenidos para las preguntas 7, 16 y 19 se muestran en las figuras 6.10.a, 6.10.b, y 6.11 respectivamente.

a) b)



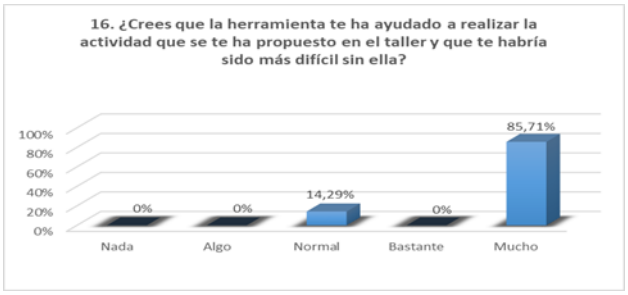


Figura 6.10. a) Respuestas a la pregunta 7. b) Respuestas a la pregunta 16.

En la Figura 6.10.a se observa que la mayoría de participantes (71,43%) creen que ILSAditor les ha resultado muy útil, mientras que al 28,57% restante les ha parecido bastante útil.

En la Figura 6.10.b se observa que la mayoría de sujetos (85,71%) piensa que la herramienta les ha ayudado "Mucho" a realizar la actividad que se les ha propuesto y que habría sido más difícil sin ella. El 14,9% restante opina que les ha ayudado "Bastante".

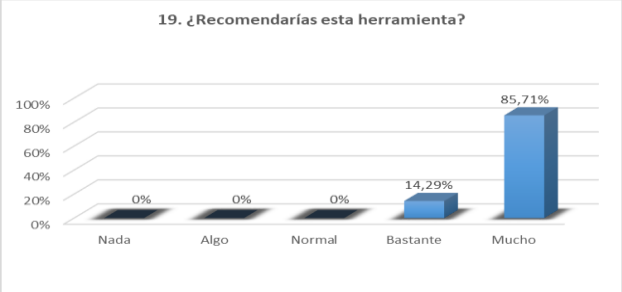


Figura 6.11 Respuestas a la pregunta 19.

En la Figura 6.11, se observa que la mayoría de participantes (85,71%) recomendarían "Mucho" la herramienta, mientras que el 14,29% restante la recomendarían "Bastante".

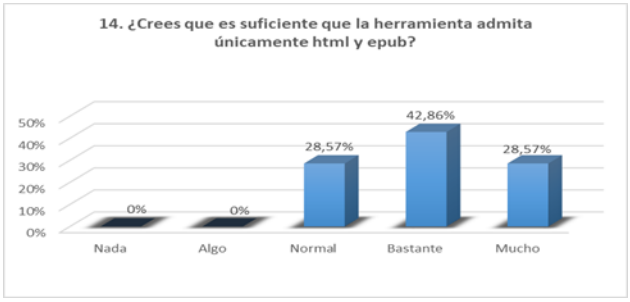
Como resumen de los resultados de las respuestas de la dimensión "utilidad de ILSAditor", la figura 6.12 muestra un gráfico circular con el promedio de las puntuaciones obtenidas en cada una de las categorías en las tres preguntas formuladas. El gráfico muestra que el 95,24% de los participantes considera muy útil o bastante útil el editor ILSAditor.

Figura 6.12 Distribución conjunta de las respuestas a las preguntas 7, 16 y 19.

4.3.5 Resultados relacionados con la mejora de ILSAditor

Por último, las cuestiones 14, 15, 17 y 18 del cuestionario, exploran la opinión de los participantes en relación a posibles mejoras que podrían realizarse sobre la herramienta. Los resultados obtenidos para las preguntas 14, 15, 17 y 18 se muestran en las figuras 6.13.a, 6.13.b, 6.14.a, y 6.14.b respectivamente.

a) b)



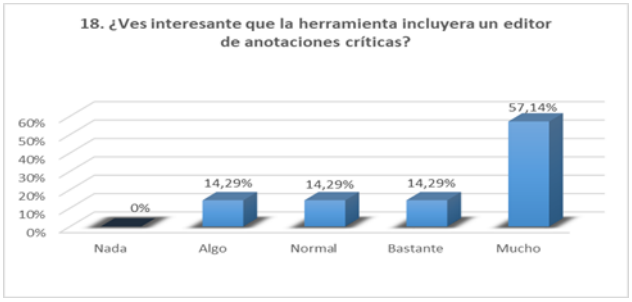
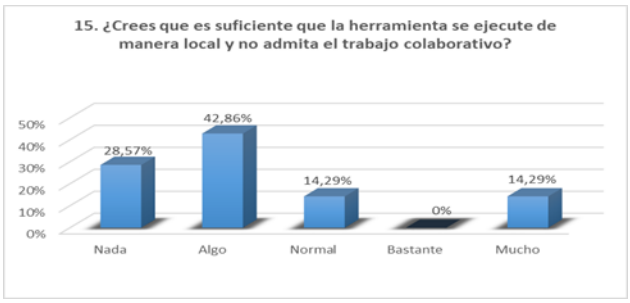


Figura 6.13. a) Respuestas a la pregunta 14. b) Respuestas a la pregunta 18.

En la Figura 6.13.a se observa que la mayoría de participantes (71,43%) creen que es suficiente con que la herramienta genere libros en formato EPUB o HTML, mientras que al 28,57% restante les ha parecido "Normal", lo cual indica, que, en general, no consideran necesario añadir otro formato a ILSAditor.

En la Figura 6.13.b se aprecia que la mayoría de los sujetos (85,71%) cree que sería interesante incluir en la herramienta un editor de anotaciones críticas, mientras que el 14,29% restante sólo lo considera algo interesante.

a) b)



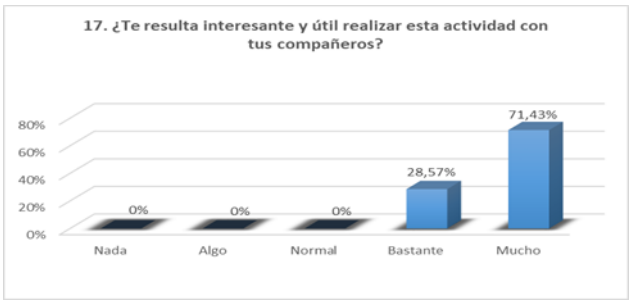


Figura 6.14. a) Respuestas a la pregunta 15. b) Respuestas a la pregunta 17.

En la Figura 6.14.a se observa que la mayoría de sujetos (71,43%) consideran que la herramienta debería permitir trabajar de forma colaborativa. El 14,9% restante opina que no lo necesitaría para trabajar con el editor.

En la Figura 6.14.b la totalidad de participantes considera muy útil realizar la actividad conjuntamente con sus compañeros.

4.4 Fiabilidad y consistencia interna del cuestionario de evaluación

Para medir la consistencia interna del cuestionario aplicado a los participantes en la experiencia de evaluación se utilizó el coeficiente alfa de Cronbach. Esta medida permite conocer la fiabilidad de un cuestionario de recogida de datos que está compuesto por preguntas cuyas respuestas están ordenadas según una escala de tipo Likert, a través de la correlación existente entre los ítems. Si el instrumento es fiable se espera que aquellos ítems que forman parte de una misma dimensión estén fuertemente correlacionadas [Welch y Comer, 1998]. Así, cuanto más cerca del valor 1 se encuentre el valor del coeficiente alfa, mayor es la consistencia interna de los ítems analizados. Se considera aceptable valores de alfa por encima de 0.7 aunque lo recomendable es un valor de alfa de 0.8 o más.

El coeficiente alfa de Cronbach obtenido a partir del cuestionario utilizado en la evaluación de la herramienta ILSAditor, arrojó un valor de 0.744, lo que representa un grado de consistencia interna por encima de lo aceptable. En la Tabla 6.1, se observa el cálculo del coeficiente del Cronbach obtenido a partir de las puntuaciones tipificadas de los diecinueve ítems del cuestionario, donde una puntuación tipificada quiere decir que la puntuación original se estandariza restando la media de las puntuaciones y posteriormente, dividiendo entre la desviación típica de la mismas.

Estadísticos de fiabilidad

Alfa de Cronbach	Alfa de Cronbach basada en los elementos (ítems) tipificados	Nº de elementos (ítems)
0,744	0,744	19

Tabla 6.1 Coeficiente de fiabilidad alfa de Cronbach.

4.5 Conclusiones obtenidas

En este capítulo se han analizado los resultados de las puntuaciones obtenidas a partir de un cuestionario de evaluación sobre la herramienta ILSAditor llevado a cabo en el contexto de un taller de formación sobre dicha herramienta. El cuestionario constaba de diecinueve preguntas agrupadas en cinco dimensiones que medían aspectos tales como la experiencia en el trabajo y en las técnicas de edición digital, la usabilidad y utilidad de ILSAditor, la utilidad de la funcionalidad del *stretchtext* y las propuestas de varias mejoras sobre el estado actual de

ILSAditor. Las conclusiones obtenidas en cada una de las dimensiones se presentan a continuación.

Con respecto a la dimensión uno (experiencia en el trabajo y las técnicas de edición digital), los resultados obtenidos muestran que, a pesar de contar con un reducido número de participantes, tan solo siete sujetos, la mayoría de ellos contaban con conocimientos previos en edición digital y con algún grado de experiencia en técnicas de *stretchtext*.

En la dimensión dos (usabilidad de ILSAditor), los resultados muestran que la mayoría de los participantes consideran que la herramienta es muy fácil de usar y tiene un diseño muy claro.

En la siguientes dos dimensiones, los resultados obtenidos revelan que la gran mayoría de los sujetos en similares proporciones, considera muy o bastante útil la utilidad del *stretchtext* y el editor ILSAditor.

Por último, la dimensión cinco, intenta medir la opinión de los participantes en relación a algunas mejoras propuestas sobre la herramienta ILSAditor. De las respuestas obtenidas, se puede concluir que la mayoría de participantes considera interesante incluir en el editor anotaciones críticas, y además creen que es suficiente que la herramienta se ejecute de manera local y admita únicamente los formatos HTML y EPUB.

Así mismo, se ha evaluado la fiabilidad y consistencia interna del cuestionario utilizado mediante el coeficiente alfa de Cronbach, habiendo obtenido como resultado un grado de consistencia interna por encima de lo aceptable.

Capítulo 5 - Conclusiones y Trabajo Futuro

5.1 Conclusiones

En este trabajo se ha analizado algunas de las iniciativas en las que se está trabajando para ampliar la funcionalidad que ofrece un libro digital con el objetivo de mejorar la experiencia e interacción de un lector de libros digitales. De todas las iniciativas analizadas, el trabajo se ha centrado en la denominada ficción interactiva que consiste esencialmente en la adicción de diferentes objetos visuales o auditivos que interaccionan con el lector con el objetivo de hacer más enriquecedora e interesante la experiencia lectora. En particular, se ha implementado la técnica del *stretchtext* que puede ser utilizada para adaptar el contenido de un libro digital a distintos tipos de lectores mediante la inclusión de niveles de visualización de los contenidos del libro digital.

Una parte fundamental de este trabajo ha consistido en probar la factibilidad técnica de las técnicas analizadas mediante el diseño y desarrollo de una herramienta software, denominada ILSAditor, que implementa la técnica del *stretchtext,* así como otros elementos propios de la ficción interactiva. La herramienta permite entre otras operaciones la apertura de documentos en formato HTML, texto plano y EPUB, la generación de *stretchtext* y la publicación de libros en formato web y dispositivos electrónicos. El diseño se ha realizado tratando de que el uso de la misma fuera lo más sencillo e intuitivo posible, sin requerir al usuario de conocimientos informáticos avanzados.

Por último, otra aportación del trabajo ha consistido en la evaluación de la herramienta mediante la realización de una experiencia de uso con profesores y estudiantes de la UCM y la UNED. Como parte de la experiencia, los participantes rellenaron un cuestionario orientada a evaluar la usabilidad de la misma, cuyos resultados se presentaron en el capítulo seis. Con la prudencia de las condiciones estadísticas en el que se realizó la experiencia (participación de 7 personas), los resultados muestran satisfacción con respecto al grado de usabilidad y la claridad del diseño de ILSAditor, así como el carácter innovador del mismo en el área de las Humanidades.

Un punto débil de este trabajo es el formato en el que se ha implemento la herramienta, una aplicación de escritorio. En este sentido, limita las posibilidades de extensión y uso de la herramienta. Así, por ejemplo, la posibilidad de realizar trabajos de edición colaborativa o el uso de recursos de la web no es posible o, en caso de serlo, es complicado. La razón de haber optado por este formato se ha debido a la posibilidad de reutilizar el editor utilizado como base de la herramienta el cual presenta una interface amigable al usuario y permite de una manera simple la extensión de sus funcionalidades para implementar las técnicas estudiadas en el trabajo. Sin embargo, tal como se comentará en la siguiente sección, futuras versiones de la herramienta deberían implementarse en formato web.

5.2 Trabajo Futuro

A partir de la realización de este trabajo, las principales líneas de trabajo se centran en ampliaciones de la funcionalidad de la herramienta desarrollada en varios sentidos.

La primera línea de trabajo planteada consistiría en convertir la herramienta actual en una aplicación web, de manera que soportará la gestión de usuarios, la definición de diferentes roles dentro del proceso de edición de un texto digital y el almacenamiento de información procedente del uso de la herramienta en un sistema de persistencia tal como una base de datos relacional.

Si la herramienta se transformara en una aplicación web, se podría plantear una segunda línea de trabajo consistente en soportar la anotación crítica colaborativa de los libros digitales que se editan. Una actividad común en los textos literarios es acompañarlos de información adicional que aparece intercala en el texto a modo de anotaciones. El objetivo de esta línea de trabajo es que la creación de estas anotaciones pudiera realizarse entre un conjunto de autores que trabajaran simultáneamente sobre el mismo texto.

Otra extensión de la herramienta en su formato web sería la explotación de la información almacenada en el sistema de persistencia con fines variados tales como facilitar el proceso de anotación realizando recomendaciones a los anotadores críticos o incluso realizando anotaciones semiautomáticas.

Por último, una tercera línea de trabajo relacionada con la anotación sería la implementación de un modelo de anotación enriquecida que no se limitase a proporcionar información adicional en formato textual y que permitiera contemplar otros tipos de información complementaria tales como información geográfica en forma de mapas de Google Maps, información multimedia mediante la inserción de videos, audios o la inserción de elementos de realidad aumentada. Estos elementos facilitarían una experiencia más enriquecedora al lector del libro digital.

Chapter 5 - Conclusions and future work

5.1 Conclusions

In this work we have analyzed some of the initiatives in which we are working to expand the functionality offered by a digital book with the aim of improving the experience and interaction of a digital book reader. Of all the initiatives analyzed, the work has focused on the so-called interactive fiction that consists essentially of the addiction of different visual or auditory objects that interact with the reader in order to make the reading experience more enriching and interesting. In particular, the *stretchtext* technique that can be used to adapt the digital book content to different reader types has been implemented by including digital book contents visualization levels.

A fundamental part of this work has been to test the technical feasibility of the techniques analyzed through the design and development of a software tool, called ILSAditor, which implements the *stretchtext* technique as well as other elements of interactive fiction. The tool allows, among other operations, open documents in HTML format, plain text and EPUB, *stretchtext* generation and publish books in web format and electronic devices. The design has been made trying to make the use of it as simple and intuitive as possible, without requiring of advanced computer knowledge by the user.

Finally, another contribution of the work has consisted in the tool evaluation through the realization of a use experience with professors and students from UCM and the UNED universities. As part of the experience, the participants filled in a questionnaire aimed at evaluating the tool usability, whose results were presented in chapter six. With the prudence of the statistical conditions in which the experience was carried out (participation of 7 people), the results show satisfaction with the degree of usability and clarity of the ILSAditor design, as well as the innovative nature of it in the area of the Humanities.

A weak point of this work is the format in which the tool has been implemented, a desktop application. In this sense, it limits the possibilities of the tool extension and use. Thus, for example, the possibility of collaborative editing works or the use of web resources is not

possible or, if it is, it is complicated. The reason for opting for this format has been due to the possibility of reusing the editor used as the basis of the tool, which presents a user friendly interface and allows in a simple way the extension of its functionalities to implement the techniques studied in the job. However, as will be discussed in the next section, future tool versions should be implemented in web format.

5.2 Future Work

As of the completion of this degree project for master, the main lines of work are focused on tool functionality extensions developed in several ways.

The first line of work proposed would be to convert the current tool into a web application, so that it will handle the management of users, definition of different roles within the process of editing a digital text and the information storage from the tool usage in a persistence system such as a relational database.

If the tool were transformed into a web application, a second line of work could be considered, consisting of handling the collaborative annotation of the digital books that are edited. A common activity in literary texts is to accompany them with additional information that appears inserted in the text as notes. The goal of this line of work is the creation of these annotations could be done among a set of authors who work simultaneously on the same text.

Another extension of the tool in its web format would be the take advantage of the information stored in the persistence system for various purposes such as facilitating the annotation process by making recommendations to critical annotators or even making semi-automatic annotations.

Finally, a third line of work related to annotation would be the implementation of an enriched annotation model that would not simply provide additional information in textual format and that would allow contemplating other types of complementary information such as geographic information in the form of maps of Google, multimedia information through the insertion of videos, audios or the insertion of augmented reality elements. These elements would facilitate a more enriching experience for the digital book reader.

Bibliografía

[A-Frame] Framework JavaScript utilizado para construir experiencias de realidad virtual, accesible online vía https://aframe.io/

[Andrews, 2004] Andrews J. (2004). On Lionel Kearns. Accesible online vía http://www.vispo.com

[AngularJS] Framework MVC de JavaScript, accesible online vía https://angularjs.org/

[Babylon] Librería utilizadas para generar animaciones y juegos, accesible online vía https://www.babylonjs.com/

[Backbonejs] Framework MVC de JavaScript, accesible online vía http://backbonejs.org/

[Blast Theory, 2003] Blast Theory (2003). Uncle Roy All Around You. Accesible online vía http://www.blasttheory.co.uk/projects/uncle-roy-all-around-you/

[Belanger y Petit, 2007] Belanger M. y Petit M.R, (2007). Disappearing Places and Time Indefinite. Accesible online vía http://turbulence.org/project/disappearing-places-and-time-indefinite/#

[Boyle C y Encarnacion A, 1994] Boyle C y Encarnación A.O. (1994). Metadoc: An Adaptative Hypertext Reading System. User Modeling and User-Adapted Interaction 4:1-94, 1994. Kluwer Academic Publishers.

[BookBlock] Librería JavaScript para simular el efecto Flipping Book, https://tympanus.net/Development/BookBlock/

[Booklet] Librería JavaScript para simular el efecto Flipping Book, http://builtbywill.com/booklet/#/

[Bootstrap] Framework de maquetación JavaScript, accesible online vía http://getbootstrap.com/

[Bootz, 2004] Bootz P. (2004). La série des U. Electronic Literature Collection, Volumen uno.

[Bray et al, 2008] Bray T., Paoli J., Sperberg-McQueen C. M., Maler E. y Yergeau F. (2008). Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C Recommendation 26 November 2008. Accesible online vía https://www.w3.org/TR/xml/

[Brusilovsky P., 1997] Brusilovsky P. (1997). Efficient techniques for adaptative hipermedia. In: Nicholas C., Mayfield J. (eds) Intelligent Hypertext. Lecture Notes in Computer Science, vol 1326. Springer, Berlin, Heidelberg.

[Brusilovsky et al, 2004] Brusilovsky P., Sosnovsky S., Shcherbinina O. (2004). E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education, 2004 in Washington, DC, USA ISBN 978-1-880094-54-9 Publisher: Association for the Advancement of Computing in Education (AACE), Chesapeake, VA

[Calameo] Plataforma web para la creación y publicación de libros y revistas digitales, https://es.calameo.com/

[Cayley, 2006] Cayley J. (2006). Lens: The Practice and Poetics of Writing in Immersive VR (A Case Study with Maquette). LEA Vol 14 Issue 05 – 06.

[Cardiff, 2000] Cardiff J. (2000). Audiowalks. Accesible online vía http://www.cardiffmiller.com/artworks/walks/

[Cardiff, 2005] Cardiff J. (2005). Her Long Black Hair. Accesible online vía https://phiffer.org/hlbh/
[cervantesvirtual] Biblioteca virtual Miguel de Cervantes, accesible online vía http://www.cervantesvirtual.com
[Chart.js] Librería JavaScript de visualización de datos, disponible online vía http://www.chartjs.org/
[Chartist.js] Librería JavaScript de visualización de datos, disponible online vía http://gionkunz.github.io/chartist-js/
[Conklin, 1987] Conklin, J. (1997). Hypertect: An Introduction and Survey. Computer 20(9), 17-41.
[d3js] Librería JavaScript de visualización de datos, disponible online vía https://d3js.org/
[De Bra et al, 2004] De Bra P., Brusilovsky P., Houben G. (2004). Adaptative Hypermedia: From Systems to Framework (2004). International Conference, AH 2000 Trento, Italy, August 28–30, 2000 Proceedings. Springer.
[De la Garza-García, 2013] De la Garza-García, J., Morales-Serrano, B.N. y González-Cavazos, B.A. (2013). Análisis Estadístico-Multivariado: Un enfoque teórico y práctico. Monterrey, México: McGrawHill.
[Dea C. et al, 2014] Dea C., Heckler M., Grunwald G., Pereda J., Phillips S. M (2014). JavaFX 8 Introduction by Exaple, Second Edition. Ed. Apress. ISBN 978-1-4302-6461-3
[Donikian et al, 2004] Donikian S., Portugal JN. (2004) Writing Interactive Fiction Scenarii with DraMachina. In: Göbel S. et al. (eds) Technologies for Interactive Digital Storytelling and Entertainment. TIDSE 2004. Lecture Notes in Computer Science, vol 3105. Springer, Berlin, Heidelberg
[ELO] Organización de Literatura Electrónica, accesible online vía https://eliterature.org/what-is-e-lit/
[EmberJS] Framework MVC de JavaScript, accesible online vía https://www.emberjs.com/
[Evernote] aplicación informática que permite la organización personal. Accesible online vía https://evernote.com/intl/es/
[Fancellu, 2017] Fancellu D. (7 diciembre 2017). Force directed graph for D3.js v4 with labelled edges and arrows. Accesible online vía http://bl.ocks.org/fancellu/2c782394602a93921faff74e594d1bb1
[Faulkner et al, 2016] Faulkner S., Eicholz A., Leithead T. y Danilo A. HTML 5.1. W3C Recommendation, 1 November 2016. Accesible online vía https://www.w3.org/TR/html51/
[Fisher, 2001] Fisher C. (2001). These Waves of Girls. Winner of the Electronic Literature Organization's 2001 Award for fiction.
[Foundation] Framework CSS JavaScript, accesible online vía http://foundation.zurb.com/
[Gamma et al, 1994] Gamma E., Helm R., Johnson R. y Vlissides J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional Computing Series.
[Garris, 2011] Garrish, M (2011). What is EPUB 3?. O'Really Media.
[Gauchat, 2012] Gauchat J.D. (2012). El gran libro de HTML5, CSS3 y JavaScript. MARCOMBO.
[GitHub] Plataforma de desarrollo de software, accesible online vía https://github.com/

[Glazkov e Ito, 2014] Glazkov D. e Ito H. Introduction to Web Components. W3C Working Group Note 24 July 2014. Accesible online vía http://w3c.github.io/webcomponents/
[Goldfarb, 1991] Goldfarb C.F. (1991). The SGML handbook. Oxford University Press, Inc.
[Google Charts] Herramienta de visualización de Google Charts, accesible online vía https://developers.google.com/chart/?utm_source=analyzo
[Hair et al, 1999] Hair, J.F, Anderson, R. E., Tatham, R. L. y Black, W. C. (1999). Análisis Multivariante, 5ª. Edición. Madrid: Prentice-Hall
[Heibach et al, 2004] Heibach C., Wenz K., Seaman B., Kac E. (2004). P0Es1S: The Aesthetics Of Digital Poetry. Hatje Cantz Publishers.
[IDPF] International Digital Publishing Forum, creadores del formato Epub. Accesible online vía http://idpf.org/epub
[Ingold, 2001] Ingold J. (2001). All Roads. Accesible online vía http://www.ifarchive.org
[ISO] Organización Internacional de Normalización, accesible online vía https://www.iso.org/home.html
[Jackson, 1995] Jackson S. (1995). Patchwork Girl. Eastgate Systems.
[JavaFX] Familia de productos y tecnologías de Oracle Corporation, accesible online vía http://www.oracle.com/technetwork/java/javase/overview/javafx-overview-2158620.html
[jfMagnify] librería JavaScript para crear efecto de lupa sobro los elementos HTML, accesible online vía https://github.com/fonstok/jfMagnify
[Joyce, 1987] Joyce M. (1987). Afternoom. Accesible online vía http://www.wwnorton.com/college/english/pmaf/hypertext/aft/
[Joyce, 1995] Joyce M. (1995). Of Two Minds Hypertext Pedagogy and Poetics. Michigan Publishing University of Michigah Press.
[Joomag] Plataforma web para la creación y publicación de libros y revistas digitales, https://www.joomag.com/es/about-us
[Hayles, 2007] Hayles N. K. (2007). Electronic Literature: What is it?. Accesible online vía https://eliterature.org/pad/elp.html
[Joomla] Sistema de gestor de contenidos, accesible online vía https://www.joomla.org/
[Kearns] Bibliografía y trabajos de Lionel Kearns, accesible online vía https://canpoetry.library.utoronto.ca/canpoetry/kearns/index.htm
[Kesteren et al, 2014] Kesteren A. V., Linss P. y Lilley C. CSS Namespaces Module Level 3. W3C Recommendation 29 September 2011, edited in place 20 March 2014. Accesible online vía https://www.w3.org/TR/css-namespaces-3/
[Landow G., 2015] Landow G (2015). Writing Poetry in Stretchtext Literature and New Form of Hypermedia. Tekka 3.2. Web.
[Lees] Preprocesador de hojas de estilo CSS, accesible online vía http://lesscss.org/
[Leishman, 2004] Leishman D. (2004). Deviant: The Possession of Christian Shaw. Accesible online vía http://www.6amhoover.com
[Likert, 1932] Likert, R (1932). A technique for the measurement of attitudes. Archives of Psychology. Número 140, pp 1-55.
[Memmott, 2000] Memmott T. (2000). Lexia to Perplexia. Electronic Literature Collection, Volume One State of the Arts. Electronic Literature Organization.

[Meteor js] Framework para automatizar y simplificar el desarrollo de aplicaciones web, accesible online vía https://www.meteor.com/

[Mitsuhara H. et al, 2001] Mitsuhara, H., Kurose, Y., Ochi, Y., Yano, Y. (2001). ITMS: Individualized Teaching Material System-adaptive integration of web pages distributed in some servers. Proceedings of EDMEDIA'2001–World Conference on Educational Multimedia, Hypermedia and Telecommunications, June 25–30, 2001. Tampere, Finland, AACE.–pp. 1333–1338

[Moodle] Gestor de contenidos educativos, accesible online vía https://moodle.org/

[Montfort, 2005] Montfort N. (2005). Twisty Little Passages: An Approach to Interactive Fiction by Nick Montfort. MIT Press, Cambridge, MA, pp. 286.

[Morgan, 1999] Morgan E.L. (1999). Electronic books and related technologies. Computers in Libraries, Vol. 19 Nº10.

[Moulthrop, 1992] Moulthrop S. (1992). Victory Garden, Eastgate Systems.

[Moulthrop, 1999] Moulthrop S. (1999). Reagan Library.

[NISO] National Information Standards Organization, accesible online vía http://www.niso.org/home/

[Oracle] Documentación de Oracle, accesible online vía http://docs.oracle.com/

[Oracle docs] Documentación del webview de javaFx, accesible online vía https://docs.oracle.com/javafx/2/webview/jfxpub-webview.htm

[Playcanvas] Librería utilizada para generar animaciones y juegos, accesible online vía https://developer.playcanvas.com/en/

[Ray, 2009] Ray, E. (2009). Learning XML, 2nd Edition, Creating Self-Describing Data. O'Really Media.

[React js] Librería JavaScript utilizada para la creación de interfaces de usuario, accesible online vía https://facebook.github.io/react/

[React VR] Librería utilizada para la creación de experiencias de VR, accesible online vía https://facebook.github.io/react-vr/

[Saas] Preprocesador de hojas de estilo CSS, accesible online vía http://sass-lang.com/

[Semantic UI] Framework CSS de JavaScript, accesible online vía https://semantic-ui.com/

[Sharan, 2015] Sharan, K. (2015). Learn JavaFX 8, Building User Experience and Interfaces with Java 8. Apress. eBook ISBN 978-1-4842-1142-7

[Short, 2002] Short E. (2002). Savoir-Faire. Short. Accesible online vía http://nickm.com/if/emshort/savoir_faire.html

[Skeleton] Framework CSS de JavaScript, accesible online vía http://getskeleton.com/#download

[Stack Overflow] Sitio web utilizada por la comunidad de desarrolladores informáticos, accesible online vía https://stackoverflow.com

[Storyspace] Primera aplicación para la creación y edición de hipertextos, accesible online vía http://www.eastgate.com/storyspace/

[Stylus] Preprocesador de hojas de estilo CSS, accesible online vía http://stylus-lang.com/

[Three.js] Librería de JavaScript utilizada para generar y animar gráficos, accesible online vía https://threejs.org/

[TinyMCE]. Editor web, accesible online vía https://www.tinymce.com/

[Turn.js] Librería JavaScript para simular el efecto Flipping Book, accesible online vía http://www.turnjs.com/

[Wardrip-Fruin, 2005] Wardrip-Fruin, N. Screen. Accesible online vía http://www.noahwf.com/screen/index.html

[w3c] Consorcio W3C, accesible online vía https://www.w3.org/

[WebKit] Motor de navegación open source, accesible online vía https://webkit.org/

[Welch y Comer, 1998] Welch, S., & Comer, J. (1988). Quantitative methods for public administration: Techniques and applications. Chicago: Dorsey Press.

[whatwg] Web Hypertext Application Technology Working Group, accesible online vía https://whatwg.org/

[WhitestormJS] Librería utilizadas para generar animaciones y juegos, accesible online vía https://github.com/WhitestormJS

[Winrar] Herramienta de compresión/descompresión de archivos, accesible online vía https://www.winrar.es/

[WordPress] Sistema de gestor de contenidos, accesible online vía https://es.wordpress.com

[Xanadu] Proyecto de hipertexto Xanadu, accesible online vía http://xanadu.com/

[Yumpu] Plataforma para la creación y publicación de libros y revistas, https://get.yumpu.com

Apéndice I – Manual de Usuario

I.1 ¿Qué es ILSAditor?

Es una aplicación de escritorio que extiende la funcionalidad del editor [Tinymce] para permitir la edición y generación de textos de ficción interactiva mediante la creación de textos por capas de profundización o también llamadas "*stretchtext*", utilizando, para lograr este fin, un sistema de creación jerárquica de texto, además de las funciones tradicionales del hipertexto. El objetivo de esta aplicación es apoyar al escritor con un mecanismo de creación y de lectura de sus textos sin tener necesidad de disfrutar de unos conocimientos previos específicos de informática. De esta manera, la edición del documento se realiza viendo directamente el resultado final (i.e. un editor WYSIWYG) y utilizando unos intuitivos iconos para aplicar los efectos de código necesarios. El resultado de la creación y edición del texto interactivo se puede exportar a un formato orientado a web (HTML5, JavaScript y CSS) o a un formato orientado a dispositivos electrónicos (formato EPUB).

I.2 ¿Qué es el *Stretchtext*?

Stretchtext son capas de profundización en las que se puede dividir un texto. Esta posibilidad de manejarse en distintos niveles de lectura o profundización, permite, por ejemplo, crear un mismo cuento para distintos niveles de comprensión o edades, o jugar con las posibilidades creativas de la ocultación y develación de texto.

I.2.1 ¿En qué consiste?

Está compuesto por dos partes:

- El **stretch** o texto que sugiere al lector el contenido que se mostrará si se clica sobre él.
- El **stretchable** o el texto que se muestra u oculta, en función de si el usuario pulsa en el texto stretch.

Por ejemplo, en el texto de la Figura I.1, se observa que la parte *stretch* está resaltada en negrita y contiene un botón de color verde. La parte *stretchable* se muestra oculta.

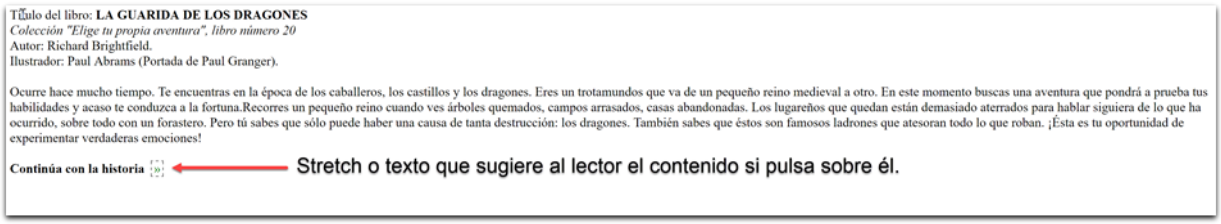


Figura I.1 Ejemplo de *stretchtext* con texto stretchable oculto.

Si se clica sobre el stretch, se muestra el contenido del *stretchable* y se cambia el icono para indicar que ya se ha expandido el texto vinculado con ese *stretch* (Figura I.2)

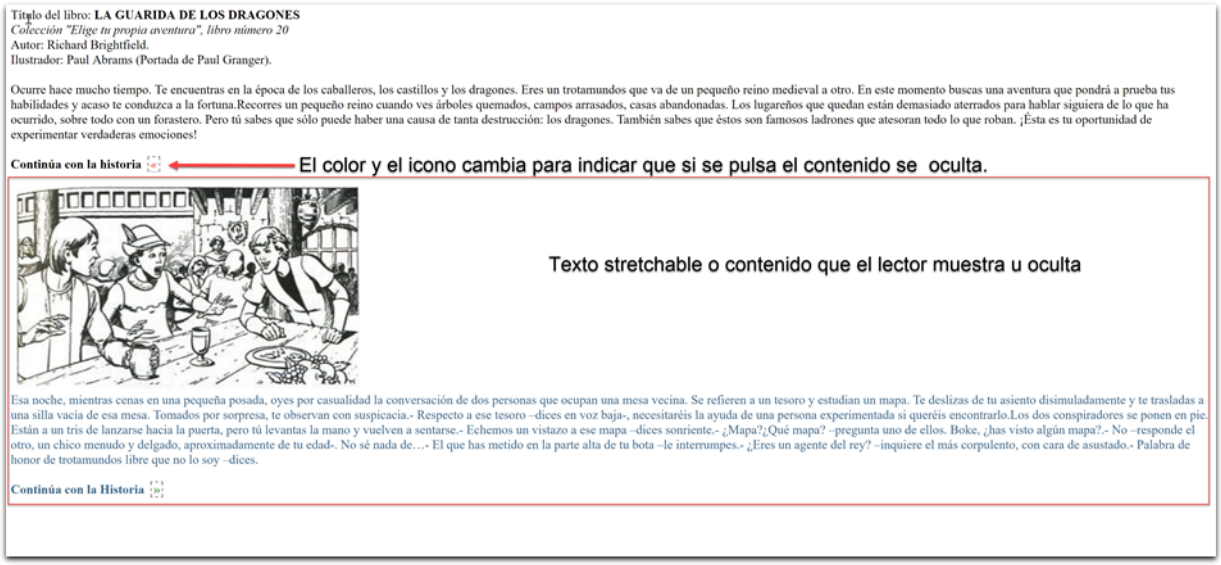


Figura I.2 Ejemplo de *stretchtext* con texto stretchable visible.

I.3 Utilización del editor

En este manual se van a explicar las opciones de ILSAditor que han sido implementadas sobre el editor [Tinymce] para permitir la utilización de la funcionalidad de *stretchtext*, que son las que forman parte de la barra de herramientas que se muestra con un recuadro rojo en la Figura I.3.

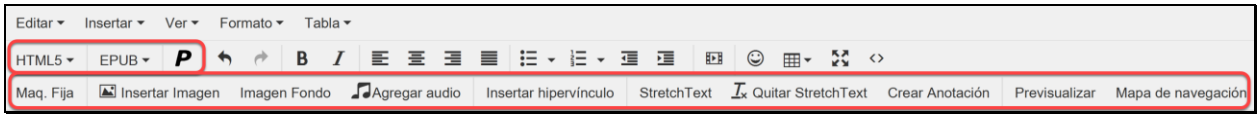


Figura I.3 Extensiones de la funcionalidad de TinyMCE.

A continuación se va a describir el funcionamiento de las trece nuevas funciones con más detalle:

I.3.1 Introducción del texto en el editor

El texto al que se le aplicará el formato *stretchtext* se puede introducir en el editor o bien escribiendo directamente en el mismo, o bien abriendo un archivo existente tanto en formato de texto plano (fichero con extensión txt), HTML o EPUB.

- Para abrir un fichero en formato EPUB, hay que clicar en el segundo botón de la barra de herramientas (“EPUB”) y escoger la primera opción (“Abrir ePub”). La Figura I.4 muestra una captura de pantalla de esta opción del menú “EPUB”.

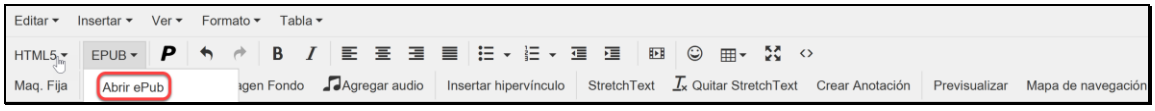


Figura I.4 ILSAditor opción Abrir ePub del menú EPUB.

A continuación, se abrirá una ventana para permitir seleccionar el fichero con el fichero EPUB que se desea trabajar (Figura I.5).

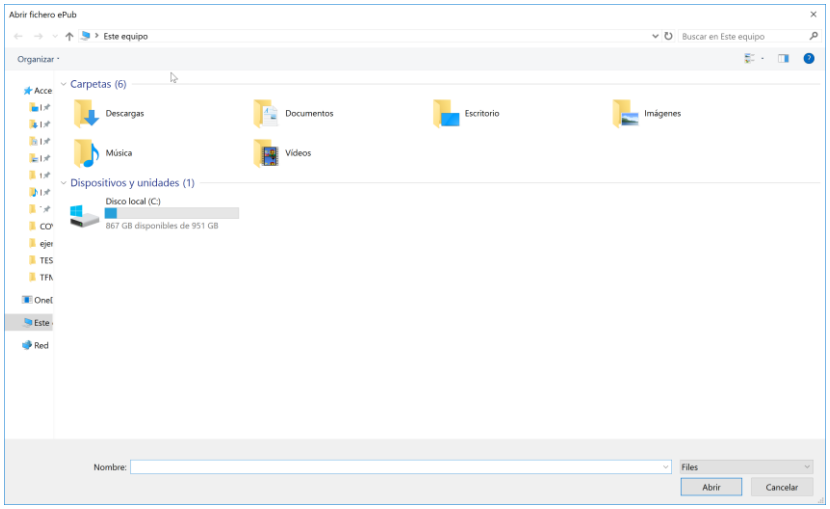


Figura I.5 Ventana de selección de apertura documento EPUB.

Una vez seleccionado el fichero, se abrirá su contenido en el editor. En la Figura I.6, se muestra un ejemplo de libro EPUB importado en ILSAditor.

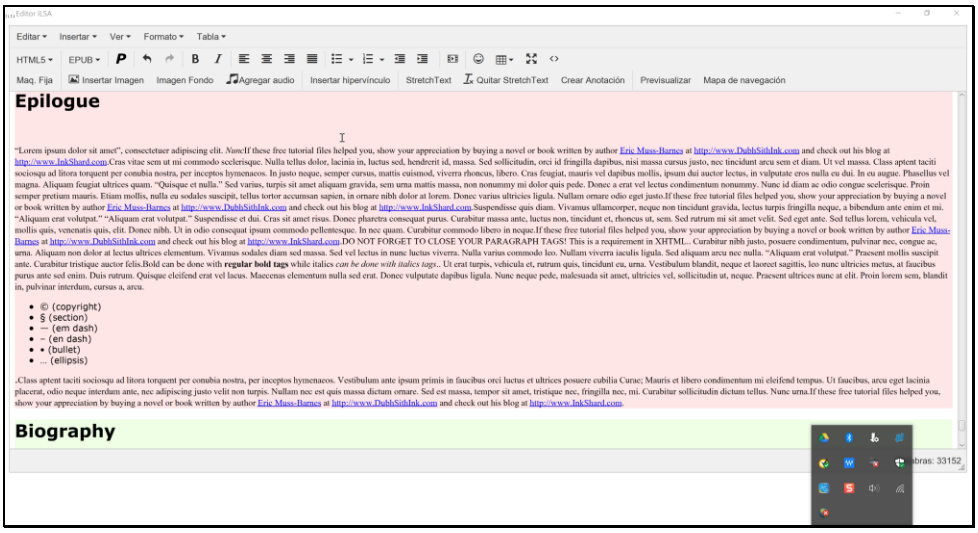


Figura I.6 Ejemplo del contenido de un archivo EPUB abierto en ILSAditor.

- Para abrir un fichero con extensión txt o HTML, hay que clicar en el primer botón de la barra de herramientas (“HTML5”) y escoger la primera opción (“Abrir archivo”). La Figura I.7 muestra una captura de pantalla de esta opción del menú “HTML5”.

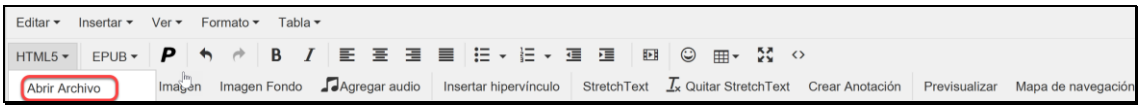


Figura I.7 ILSAditor opción Abrir Archivo del menú HTML5.

A continuación, se abrirá una ventana para permitir seleccionar el fichero con el que se desea trabajar. Los formatos permitidos son: fichero de texto (txt) o página web (HTML). Una vez seleccionado el fichero, se abrirá su contenido en el editor. En la Figura I.8, se muestra un ejemplo de libro HTML importado en ILSAditor.

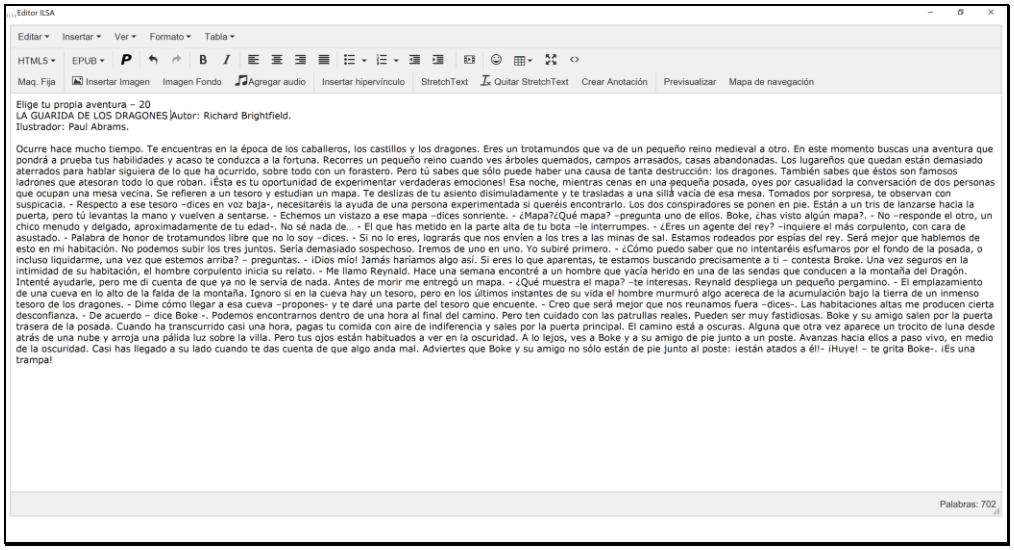


Figura I.8 Ejemplo del contenido de un archivo HTML abierto en ILSAditor.

I.3.2 Creación de texto stretchtext

La edición del texto iterativo se puede realizar mientras se está escribiendo el texto, o bien, cuando ya se ha escrito. El proceso de creación de *stretchtext* es el mismo para los dos formatos de libro que permite generar el editor, es decir, formato web (HTML5, JavaScript y CSS) y EPUB.

I.3.2.1 stretchtext de primer nivel

En este caso, se va a explicar la creación del *stretchtext* una vez se ha escrito el texto, a través del texto de ejemplo de la Figura I.9.

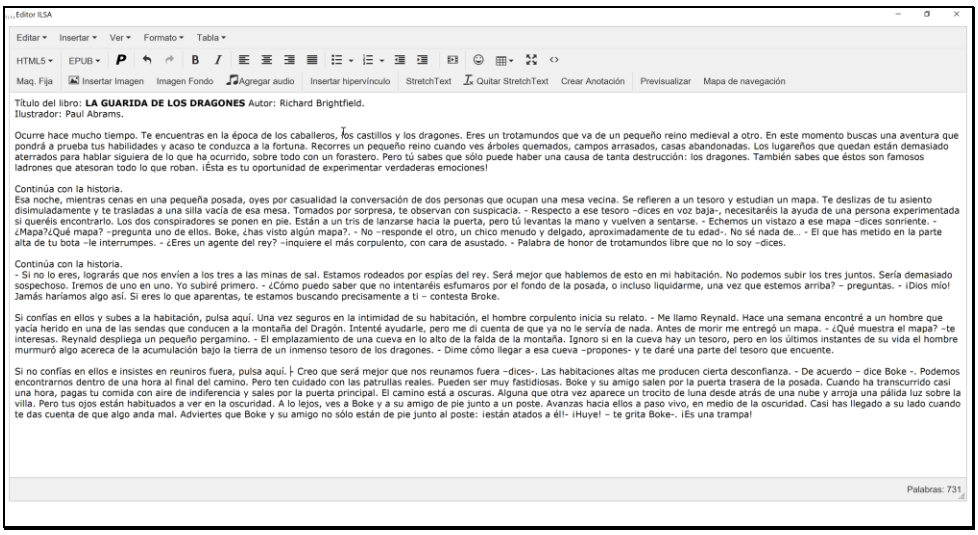


Figura I.9 Texto de ejemplo de ILSAditor.

Para crear el *stretchtext* se debe partir del texto que va a tener el primer nivel de ocultación/visibilidad. Como se puede ver en la Figura I.10, se selecciona texto desde el tercer párrafo hasta el final y, a continuación, se pulsa en el botón de la segunda barra de herramientas llamado "*StretchText*"

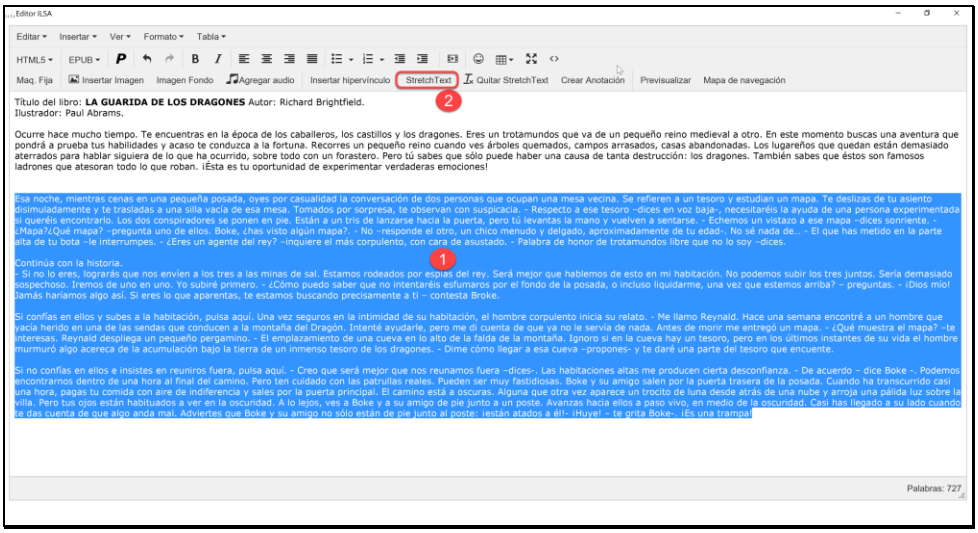


Figura I.10 Ejemplo de creación de *stretchtext* de primer nivel.

Una vez creado el *stretchtext*, el editor lo resaltará visualmente tal y como se haya implementado el selector ".stretchStyle1" en la hoja de estilos "stretch.css" ubicada en la carpeta ILSA_editor_docs\stretchEditor\style. Por defecto, tiene background-color: #EBF1F5, es decir, un color de fondo azul claro, tal y como se puede ver en la Figura I.11.

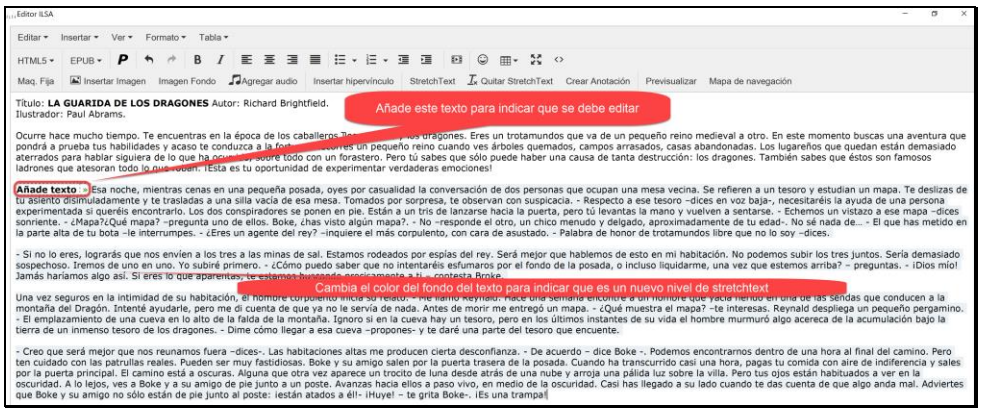


Figura I.11 Ejemplo de *stretchtext* de primer nivel.

Para terminar con la creación del *stretchtext*, únicamente tenemos que introducir un texto que sugiera al lector el contenido que se mostrará si se pulsa sobre el stretch. Para editar el stretch, se tiene que modificar el texto "Añade texto" con el texto que se desee, por ejemplo, con la frase "Continúa la historia".

I.3.2.2 Previsualización o comprobación de la funcionalidad

Si se desea comprobar el resultado de la edición del *stretchtext*, se puede previsualizar el libro pulsando en el botón "Previsualizar" de la segunda barra de herramientas, tal y como se puede ver en la Figura I.12.

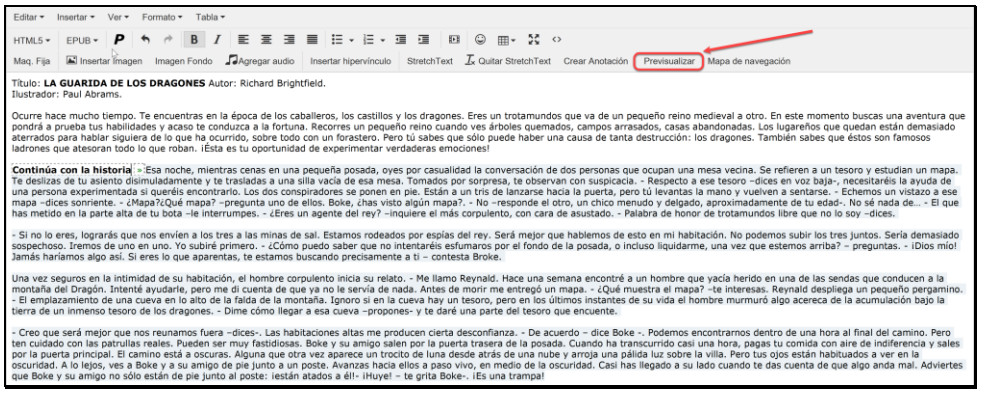


Figura I.12 Ejemplo de previsualización de texto con *stretchtext* de primer nivel.

A continuación, se abre una ventana del navegador mostrando el resultado final (Figura I.13):

Figura I.13 Ejemplo de previsualización en el navegador de *stretchtext* de primer nivel, con la parte stretchable oculta.

Una vez que se pulsa sobre el stretch, se expande el resto de contenido (Figura I.14)

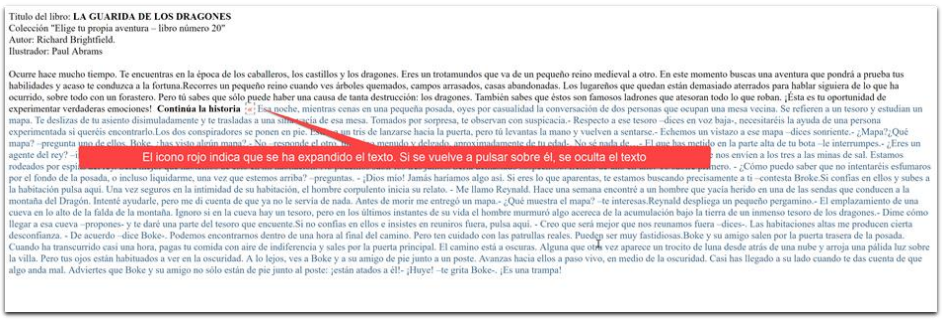


Figura I.14 Ejemplo de previsualización en el navegador de *stretchtext* de primer nivel, con la parte stretchable oculta.

I.3.2.3 Stretchtext de segundo nivel

Es importante tener en cuenta que, para crear un segundo nivel *siempre se tiene que hacer dentro del primer nivel*. Esto se va a saber, porque como se ha visto en el ejemplo anterior, el editor permite aplicar un formato personalizado al texto que pertenece a un determinado nivel. En el formato que viene por defecto el color del fondo del texto de primer nivel es de color azul claro. Para crear un segundo nivel de *stretchtext*, se debe seleccionar el texto sobre el que se quiere crear un segundo nivel y, a continuación, se pulsa en el botón *stretchtext* (Figura I.15).

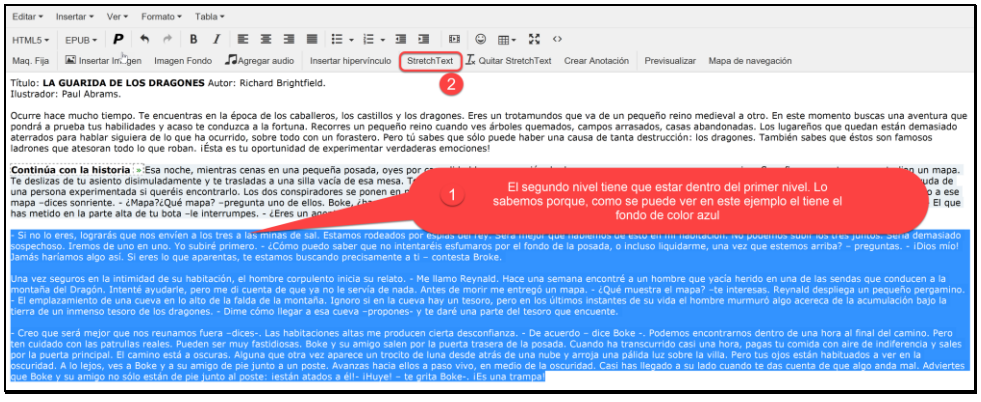


Figura I.15 Ejemplo de creación de *stretchtext* de segundo nivel.

Si se pulsa en el botón de previsualizar se puede ver su funcionamiento. A continuación, se va a explicar el funcionamiento a través de capturas de pantallas de todo el proceso representado por las Figuras I.16, I.17 e I.18.

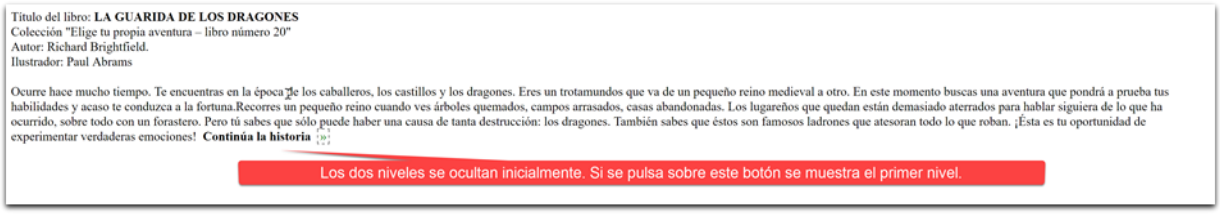


Figura I.16 Ejemplo de previsualización en el navegador de *stretchtext* de segundo nivel, con la parte stretchable de primer nivel oculta.

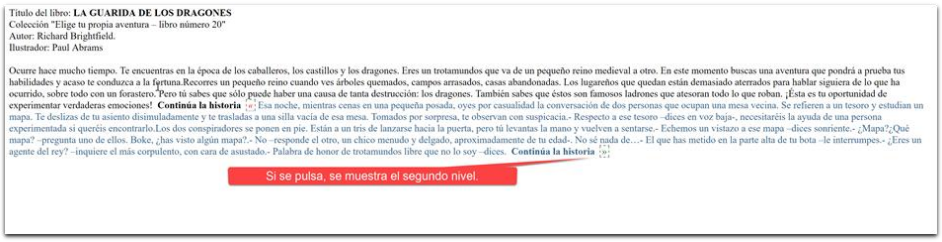


Figura I.17 Ejemplo de previsualización en el navegador de *stretchtext* de segundo nivel, con la parte stretchable de segundo nivel oculta.

Figura I.18 Ejemplo de previsualización en el navegador de *stretchtext* de segundo nivel, con los stretchables de primer y segundo nivel visibles.

I.3.2.4 Más niveles de stretchtext

En este ejemplo, se va a crear un *stretchtext* de tercer nivel. Para ello, se selecciona el texto contenido en el segundo nivel, como se puede ver en la Figura I.19.

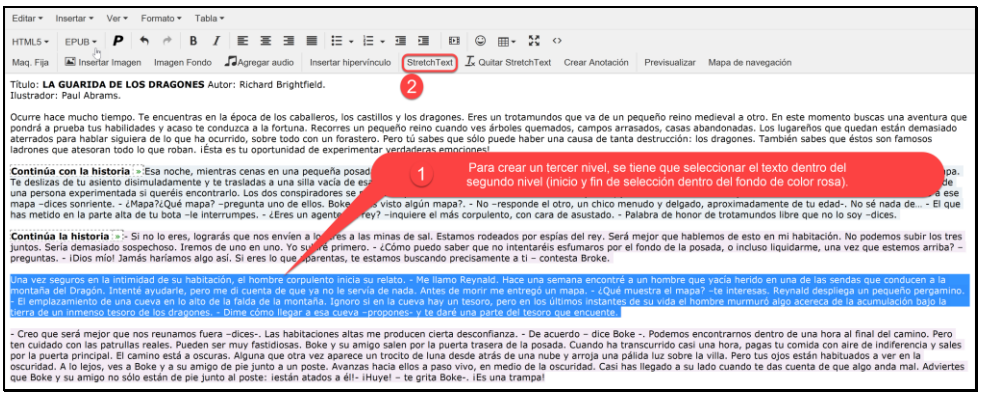


Figura I.19 Ejemplo de creación de primer *stretchtext* de segundo nivel.

En la Figura I.20, se muestra el resultado del paso anterior en el editor. Como se puede observar, por defecto, el fondo de cada nivel de *stretchtext* se identifica con colores distintos.

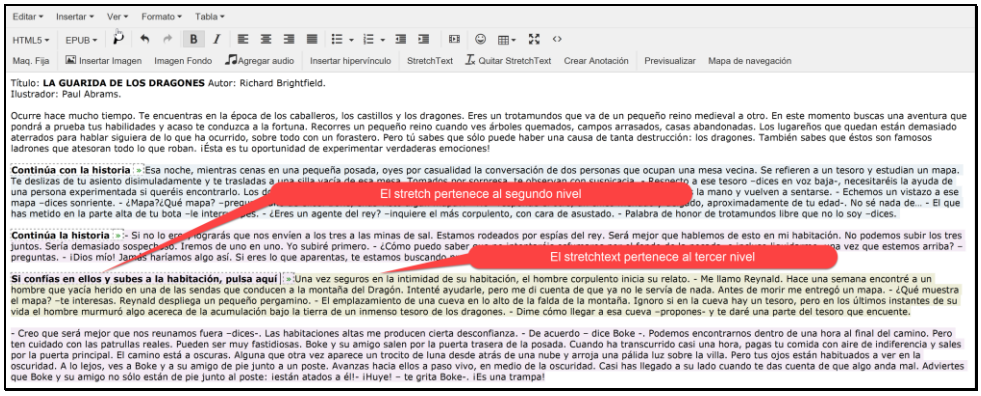


Figura I.20 Ejemplo de creación de *stretchtext* de segundo nivel.

I.3.3 Eliminación de texto stretchtext

Para eliminar texto que ha sido marcado como *stretchtext*, hay que seguir los siguientes pasos:

1. Seleccionar el texto desde antes del texto stretch (o texto que sugiere al lector el contenido que se mostrará si se pulsa) hasta el final del nivel o del stretchable (se sabrá porque tendrá otro formato).
2. Clicar sobre el botón “QuitarStretchText” de la barra de herramientas.

Las Figuras I.21 e I.22, que se muestran a continuación, ilustran los pasos que hay que seguir:

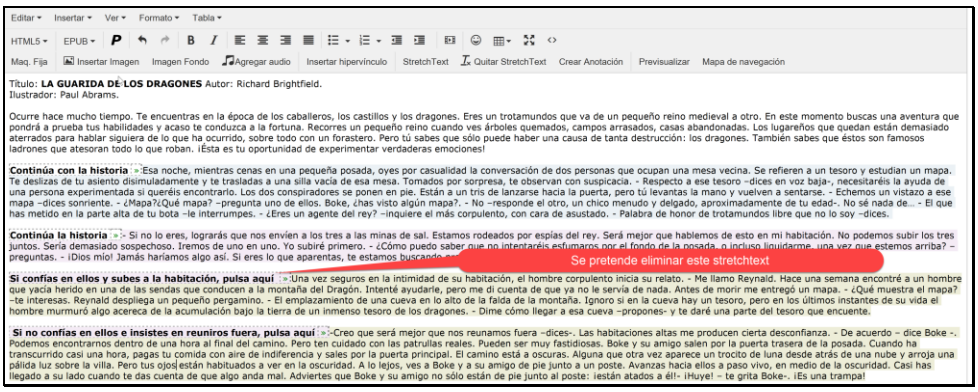


Figura I.21 Ejemplo de *stretchtext* a eliminar.

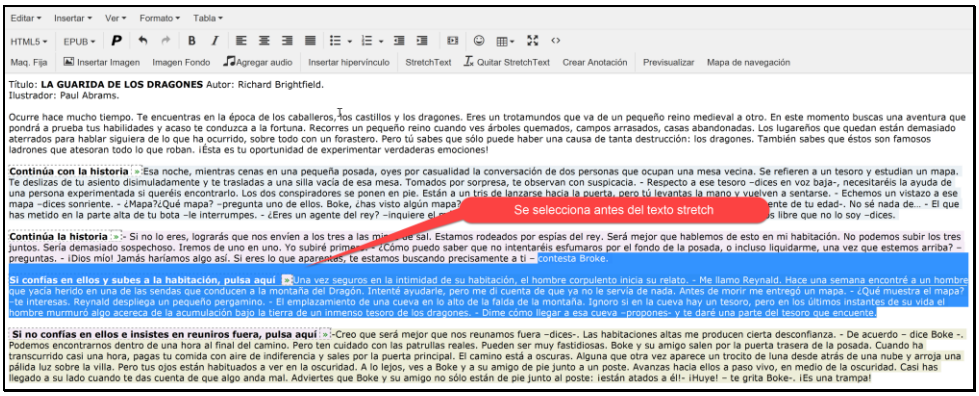


Figura I.22 Ejemplo de procedimiento de eliminación de *stretchtext*.

Como se puede comprobar en la Figura I.23, se ha eliminado el *stretchtext* de ese párrafo:

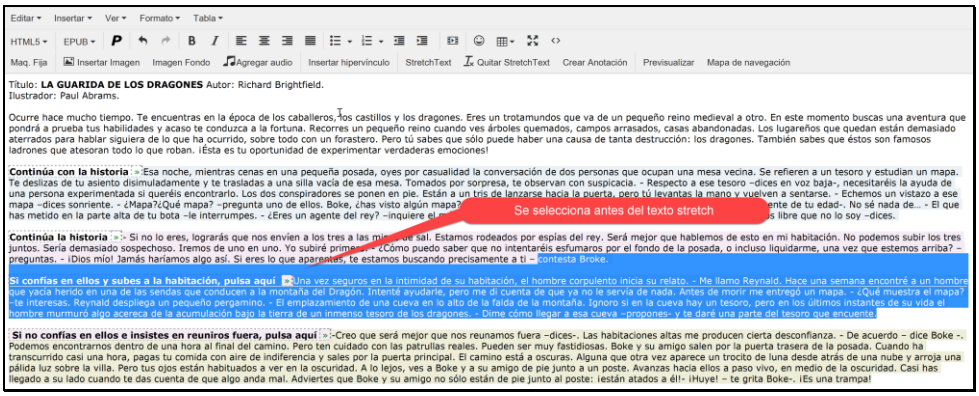


Figura I.23 Ejemplo de *stretchtext* eliminado.

I.3.4 Insertar imágenes en un nivel de stretchtext

Si se quiere añadir una imagen, por ejemplo, al principio del primer nivel de *stretchtext*, sólo hay que posicionar el puntero del ratón en el lugar en donde se quiere insertar y a continuación pulsar en el primer botón de la segunda barra de herramientas del ILSAditor, llamado “Insertar Imagen”.

A continuación, se describe gráficamente los pasos que hay que seguir para insertar una imagen a través de las Figuras I.24 e I.25:

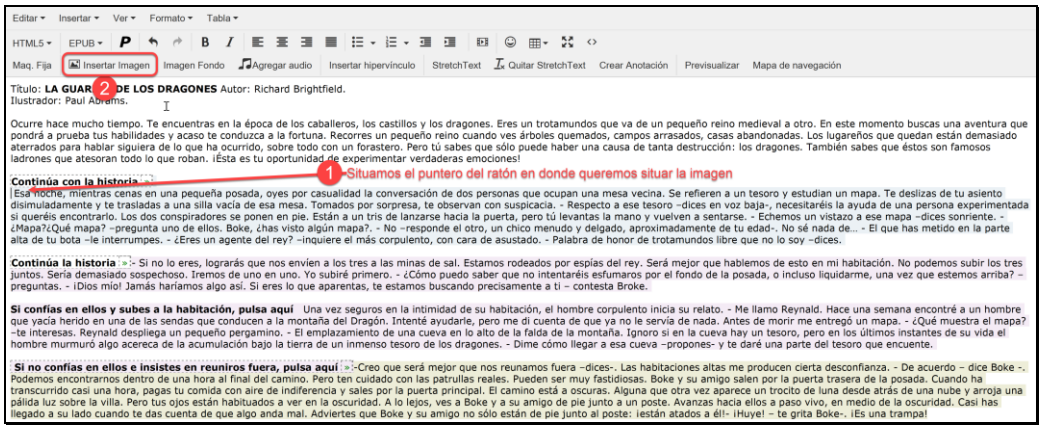


Figura I.24 Ejemplo de procedimiento de inserción de una imagen en un nivel de *stretchtext*.

A continuación, se muestra la ventana emergente que se puede ver en la Figura I.25, para poder introducir los valores de las siguientes propiedades de una imagen:

- Introduce la dirección de la imagen: Se puede introducir directamente la URL de la imagen o seleccionar una imagen del sistema de ficheros local a través del botón “Buscar”.
- Descripción emergente (tooltip): Mensaje que se muestra en el fichero HTML generado cuando se posiciona el puntero del ratón sobre la imagen.
- Espacio vertical: Espacio vertical en píxeles, es decir, posición vertical en la que se quiere ubicar la imagen.
- Espacio horizontal: Viene en píxeles. Es la posición horizontal en la que se quiere ubicar la imagen
- Posición de la imagen: Se refiere a la posición de la imagen en relación al texto:

- Seleccionada en el editor: La imagen se muestra aislada del texto.
- Derecha del texto: El texto se sitúa a la izquierda de la imagen
- Izquierda del texto: El texto se sitúa a la derecha de la imagen

- Enlazar con la URL: Permite añadir un enlace sobre la imagen con la que se está trabajando. Se puede añadir directamente la URL o seleccionar un documento del sistema de ficheros local.

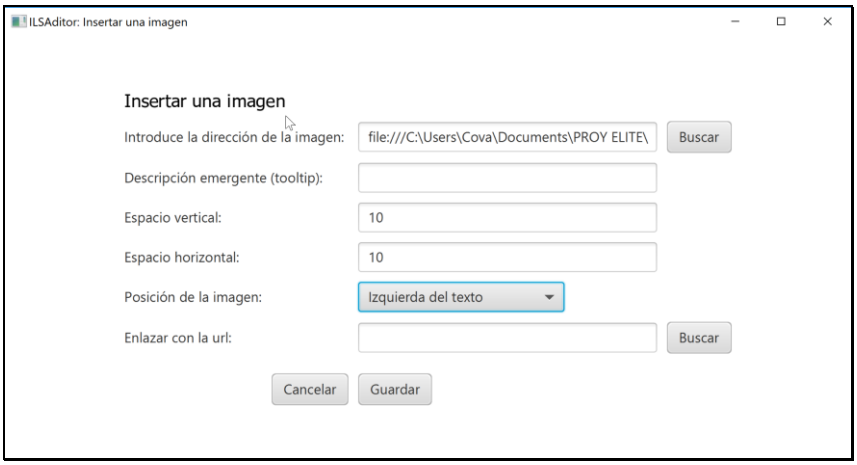


Figura I.25a Ventana emergente de inserción de una imagen.

A continuación, en la Figura I.25b, se muestra la imagen en el área de texto del editor con los valores introducidos en la Figura I.25a:

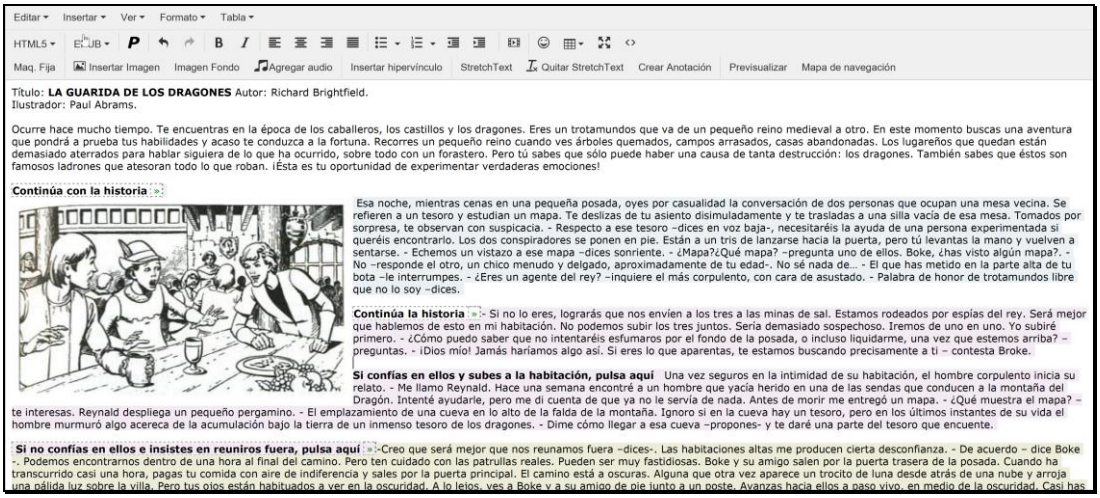


Figura I.25b Ejemplo de resultado de la inserción de una imagen en un nivel de *stretchtext*.

I.3.5 Insertar audio

Para agregar audio, se debe seleccionar el elemento (imagen, botón o texto *stretchtext*) que se quiere utilizar para reproducir al clicar sobre él. A continuación, se debe pulsar en el cuarto botón de la segunda barra de herramientas para escoger el fichero de audio que se desee insertar en el libro (Figura I.26).

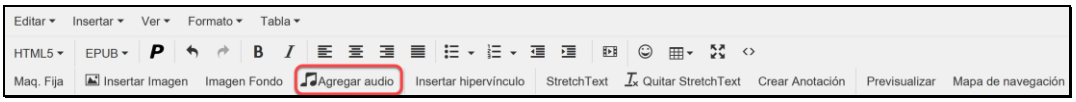


Figura I.26 ILSAditor: ubicación del botón "Agregar audio".

Una vez que se haya escogido el fichero MP3, en el editor no se mostrará nada, pero cuando se genere la página HTML, al pulsar sobre el elemento escogido, se reproducirá al clicar sobre él.

I.3.6 Insertar imagen de fondo

Para agregar una imagen de fondo, se debe posicionar el cursor del ratón en la primera posición del área de texto del editor y, a continuación, clicar en el segundo botón de la segunda barra de herramientas, llamada "Imagen Fondo" (Figura I.27).

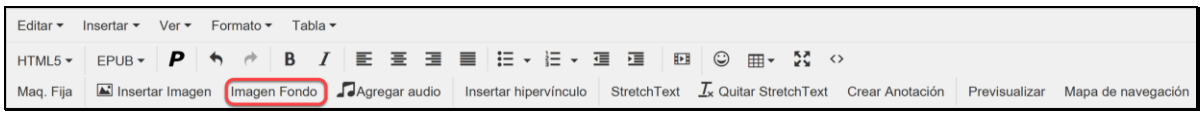


Figura I.27 ILSAditor: ubicación del botón "Imagen Fondo".

En el editor se mostrará (en la posición inicial del área de texto) una imagen que permitirá el redimensionamiento. Será en el documento generado en donde se mostrará como imagen de fondo.

I.3.7 Aplicar formato párrafo

Se puede aplicar un formato personalizado al párrafo mediante el tercer botón de la primera barra de herramientas (Figura I.28).

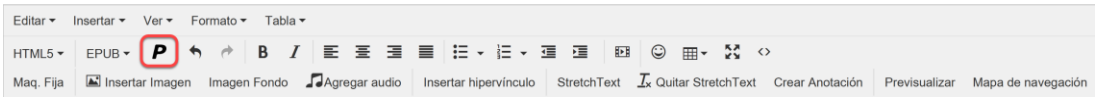


Figura I.28 ILSAditor: ubicación del botón de formato de párrafo.

La configuración del formato se puede realizar en la hoja de estilos stretchDocGen/style/strechDocGen.css en la etiqueta

I.3.8 Creación de texto enriquecido

La creación de texto enriquecido o anotaciones es sólo compatible con HTML5. Para crearlo, en primer lugar, hay que seleccionar el texto que se quiere anotar, como se puede ver en la Figura I.29.

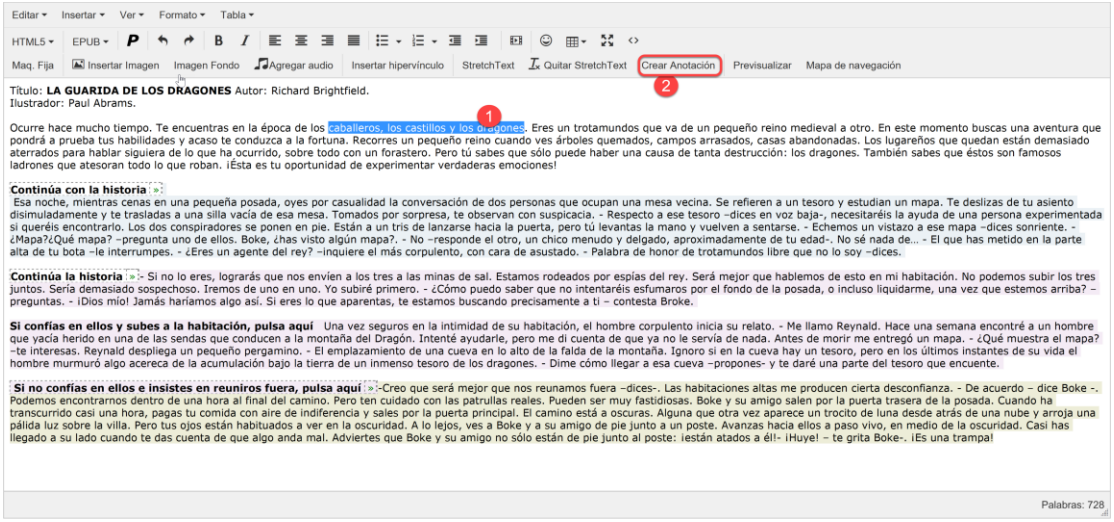


Figura I.29 ILSAditor: Selección de texto anotado.

Una vez pulsado el botón “Crear anotación”, se muestran los elementos que componen la anotación (Figura I.30)

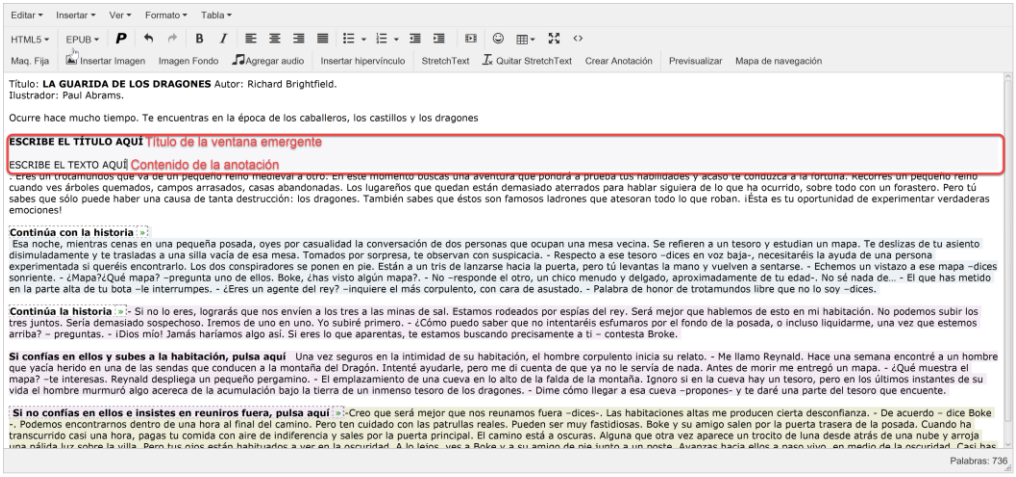


Figura I.30 ILSAditor: Componentes de una anotación.

A continuación, se muestra en la Figura I.31 un ejemplo de anotación:

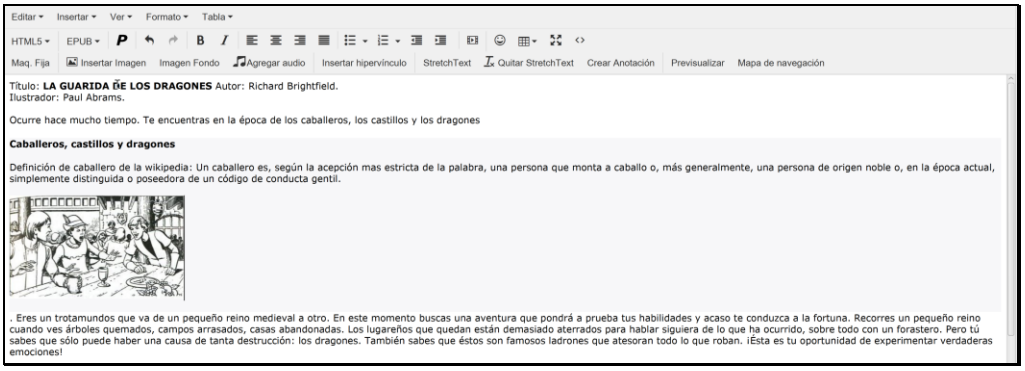


Figura I.31 ILSAditor: Ejemplo de anotación.

En las siguientes figuras (I.32 e I.33) se explica cómo se visualiza una anotación de ejemplo:

Figura I.32 ILSAditor: Visualización del texto anotado.

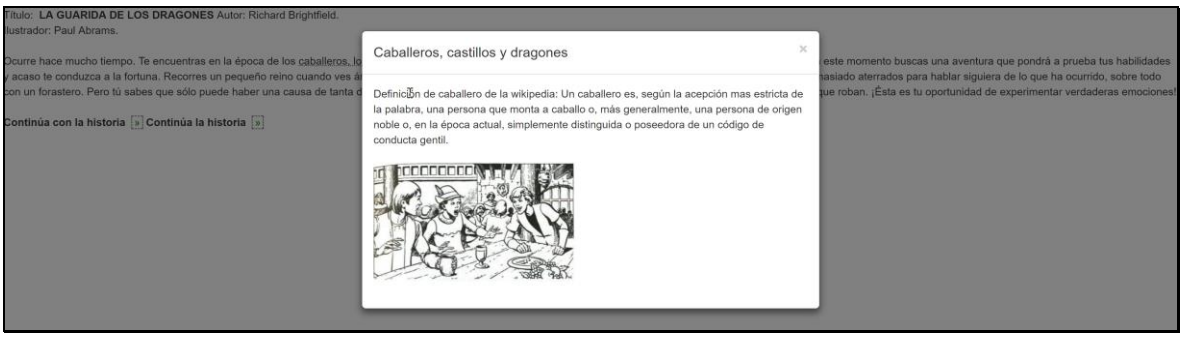


Figura I.33 ILSAditor: Visualización de la anotación.

I.3.9 Maquetación fija

La maquetación fija se puede realizar sobre un texto existente o con un documento en blanco. En ambos casos simplemente hay que pulsar en la opción “Maq. fija”. Una vez pulsada esta opción, en el área de texto se mostrará un marco de color gris con las dimensiones de una pantalla de una Tablet. La edición del texto se realizará sobre este marco (Figura I.34).

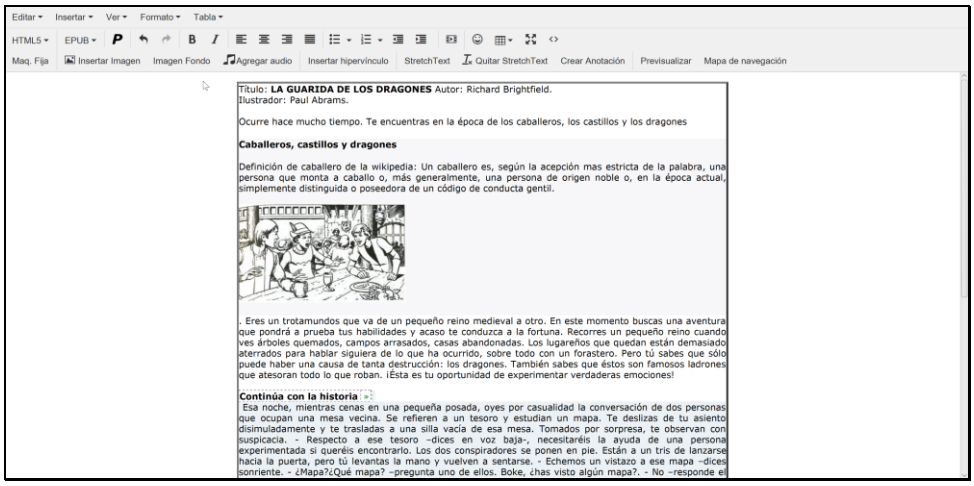


Figura I.33 ILSAditor: Ejemplo de maquetación fija.

Una vez finalizada la edición de la obra, los documentos generados tendrán una maquetación fija (sin adaptación para otros dispositivos).

I.3.10 Visualización del mapa de navegación

Para visualizar el mapa de navegación hay que pinchar sobre la opción “Mapa de Navegación” del menú del editor. A continuación, hay que seleccionar el directorio del sistema de ficheros local que se quiere mostrar.

Una vez seleccionado se mostrará las páginas web como nodos en el mapa y las relaciones (vínculos entre ambas) serán las aristas dirigidas, tal y como se puede ver en la Figura I.35.

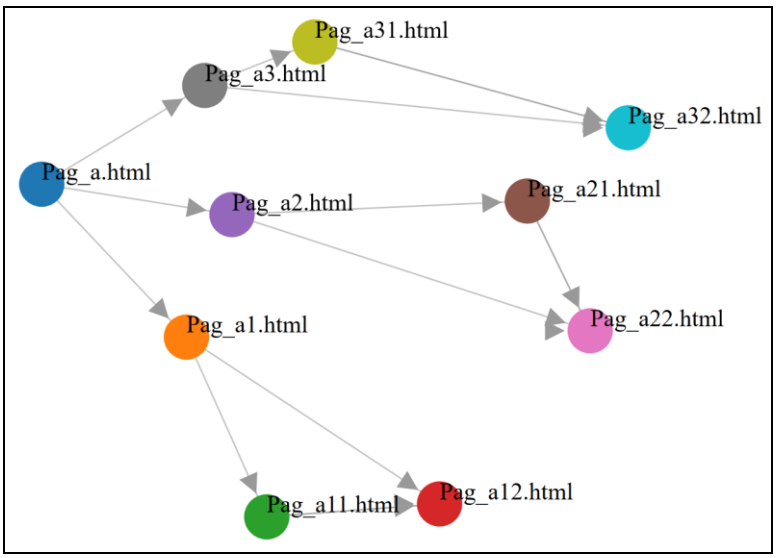


Figura I.35 ILSAditor: Ejemplo de mapa de navegación.

I.3.11 Publicación del documento

El contenido del editor se puede guardar como una página web (página HTML) comprimida en un fichero Zip o como libro en formato EPUB.

I.3.11.1 Generar el libro en formato HTML

Para crear el libro en formato HTML5 y todos sus ficheros, hay que pulsar en la barra de herramientas el menú de HTML5 y, posteriormente, seleccionar la opción “Guardar libro” (Figura I.36).

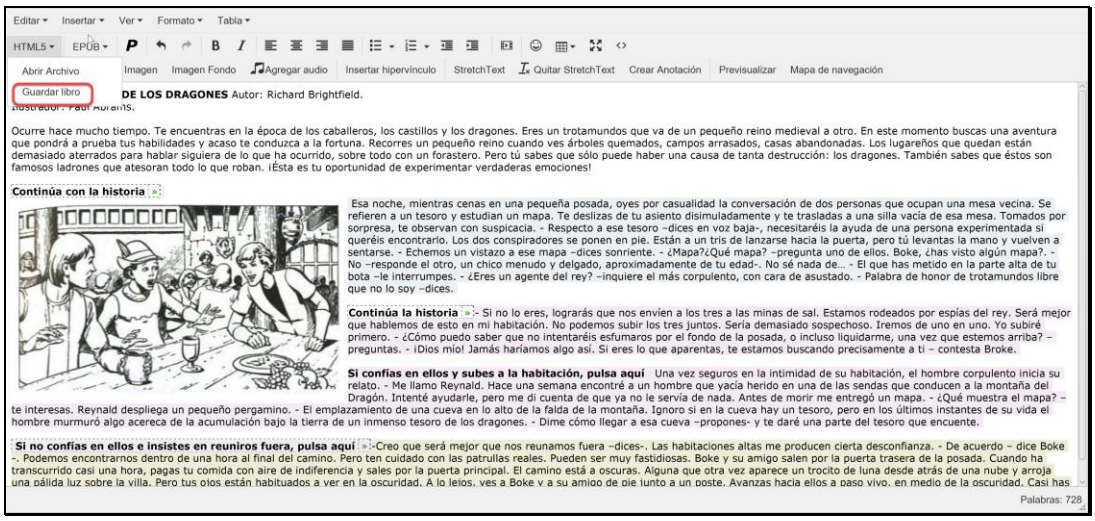


Figura I.36. ILSAditor: Ubicación de la opción "Guardar libro" del menú HTML5.

A continuación, se abrirá una ventana para seleccionar la carpeta en la que se quiere guardar el fichero comprimido e introducir el nombre del libro. Una vez seleccionado, se debe pulsar en el botón "Guardar" y, es en ese momento, cuando se genera el fichero Zip. Es necesario descomprimirlo para poder visualizarlo (por ejemplo, con una herramienta de descompresión como Winrar [Winrar]).

Como se puede ver a continuación, el contenido del libro consta de todos los documentos necesarios para el correcto funcionamiento de la página web (Figura I.37):

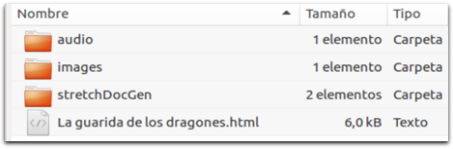


Figura I.37 Ejemplo de la estructura de ficheros que componen un libro web.

- Una página HTML: Contiene el texto con el que se ha estado trabajando.
- Una carpeta llamada Audio con todos los ficheros de audio del libro.
- Una carpeta llamada Images con todas las imágenes del libro.
- Una carpeta llamada stretchDocGen con las siguientes carpetas:

- js: con los ficheros de JavaScript jquery-3.1.1.min.js y stretch.text.js, necesarios para que el *stretchtext* funcione correctamente.
- stretchDocGen.css: Hoja de estilos del libro generado.

I.3.11.2 Generar libro en formato EPUB

Para generar el libro en formato EPUB se tiene que pulsar en el botón "EPUB" de la barra de herramientas (Figura I.38).

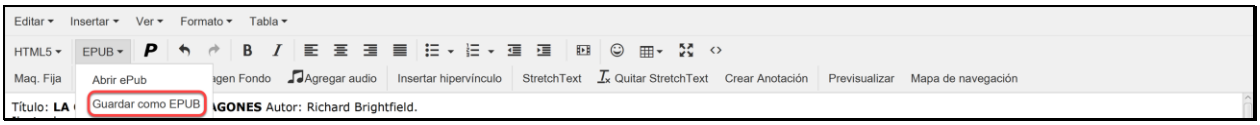


Figura I.38 ILSAditor: Ubicación de la opción "Guardar como EPUB" del menú EPUB.

El editor lanzará una ventana preguntando si deseas guardar el libro en formato EPUB. Si se pulsa que no, se descarta la creación del libro. Si se pulsa que sí, se continúa con el proceso y abrirá otra ventana informativa indicando que se debe introducir un nombre de capítulo. Se pulsa en aceptar y, como se puede comprobar a continuación, en la Figura I.39, en la primera línea del área de texto del editor, es en donde se tiene que escribir el nombre del capítulo.

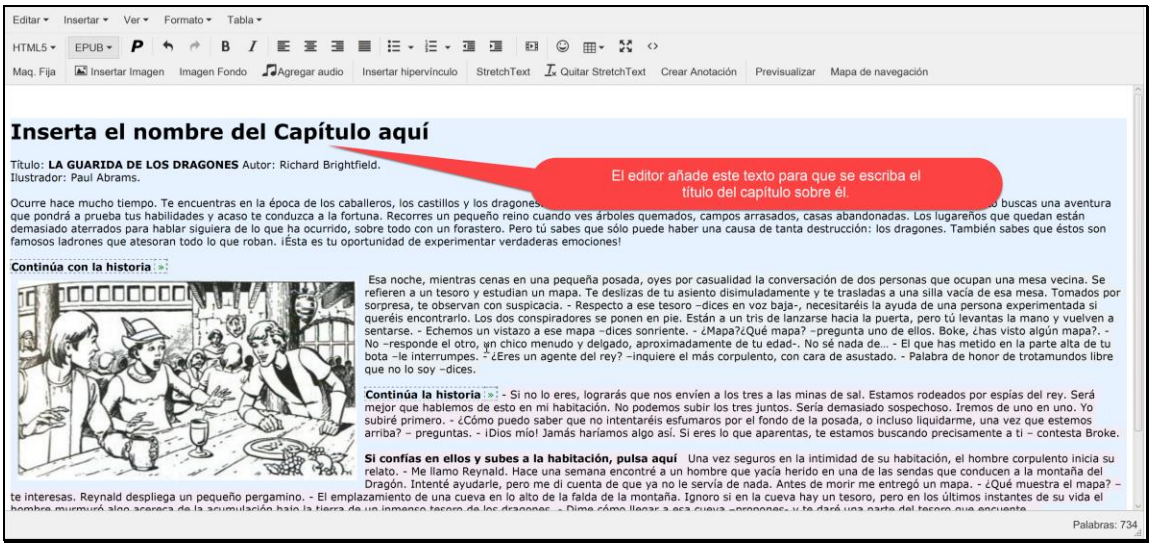


Figura I.39 Ejemplo inserción del nombre de capítulo de un libro EPUB.

Una vez introducido el nombre, lo único que queda pendiente es volver a pulsar en el menú EPUB, opción “Guardar como” y seleccionar el nombre y ubicación del libro para guardar y generar el libro.

I.4 Creación de libro EPUB

A continuación, se va a explicar qué contiene el menú EPUB y cómo crear un libro en este formato.

I.4.1 Menú EPUB

Tal y como se puede ver en la imagen a continuación, en la Figura I.40, el menú se ubica en la esquina superior izquierda:

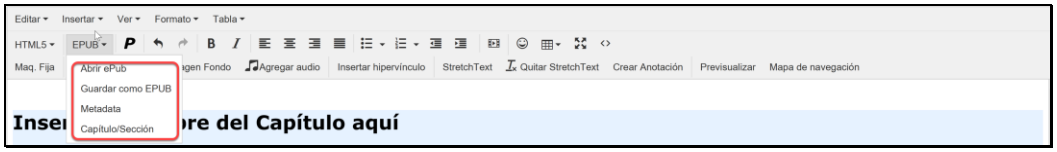


Figura I.40 ILSAditor: opciones del menú EPUB.

Una vez que se pulsa en la opción “EPUB”, se abrirá todas las opciones relacionadas con este formato.

I.4.2 Creación de un libro EPUB

Como se ha visto en el apartado anterior, en el menú EPUB, se encuentran las secciones en las que se divide un libro. Para crear una sección, hay que pulsar sobre la opción “Capítulo/Sección”. **Hay que tener en cuenta que el orden en el que se muestra en la pantalla, es el orden que se visualizará en el EPUB generado**.

Una vez creada una sección, se muestra la información relacionado con esta, siendo: el encabezado con etiqueta H1 con el texto “Inserta un capítulo/sección” que se debe modificar y el texto “Inserta el contenido del capítulo/sección aquí”, que se tendrá que sustituir por el contenido multimedia que se desee visualizar.

I.4.3 Edición de un libro EPUB

Para editar un libro existente, lo primero que se debe hacer es abrirlo en el editor para poder acceder al texto. Para ello, hay que pulsar en el menú EPUB y, posteriormente, en la opción "Abrir EPUB".

I.4.4 Previsualización de un libro en formato EPUB

La previsualización de un libro EPUB difiere del libro en formato HTML5 en que, al estar dividido en secciones independientes, la previsualización se tiene que hacer por las secciones que contenga el libro.

Por ejemplo, en el que se está editando un libro con varios capítulos, se lanzará una ventana para preguntar qué sección de las dos que contiene el libro, se desea visualizar. Para previsualizar una, hay que seleccionar de la lista la que se desee y, finalmente, se pulsa en el botón "Mostrar", para abrir una versión preliminar del libro en el navegador.

I.5 Hojas de Estilo

En las secciones anteriores se ha ido comentando que la aplicación contiene dos hojas de estilo, que son stretchDocGen\style**stretchDocGen.css** y ILSA_editor_docs\stretchEditor\style**stretch.css**.

I.5.1 Hoja de estilos del editor: stretchDocGen.css

El aspecto o la presentación del contenido del área de texto del editor se puede configurar modificando esta hoja de estilos. A continuación, se va a explicar su contenido:

```
.stretchblock:after{    /*Contenedor de los componentes de stretchText */
padding-left: 3px;
padding-right: 3px;
display:inline;
}
```

```
        .stretchblock.hidden:after{
    padding-left: 3px;
    padding-right: 3px;
    display:inline;
}

        .stretchable {  /* Parte o contenido que se muestra  u oculta del stretchText */
            text-decoration: none;
    • webkit-transition: opacity 0.2s ease-out;
    • moz-transition: opacity 0.2s ease-out;
    • o-transition: opacity 0.2s ease-out;
        transition: opacity 0.2s ease-out;
        display:inline;
}

        .stretch { /* texto que sugiere al lector el contenido que se mostrará si se clica sobre él.
*/
    cursor: pointer;
    color: inherit;
    text-decoration: none;
    display:inline;
}

        .stretch:after{  /* Formato antes del stretch. Borde punteado e icono de color verde */
    border: 1px dashed gray;
    content: "\00bb";
    color: green;
    padding-left: 3px;
    padding-right: 3px;
    display:inline;
```

```
}
        .stretch.hidden:after{   /*  Formato antes del stretch cuando se muestra. Borde punteado e icono de color rojo */
    border: 1px dashed gray;
    content: "\00ab";
    color: red;
    border-bottom:1px dotted #9999CC;
    text-decoration:none;
    font-style:italic;
    display:inline;
}
        div.stretchblock, div.stretch, div.stretchable{   /* Muestra los stretchblock, stretch y stretchable como un elemento inline (como un span)*/
            display: inline;
}
        div.p{ /* muestra div.p como un span*/
            display: inline;
}
        .stretchStyle1{ /* A continuación se formatean los niveles o capas de profundidad de stretchtext. Si se añaden más de 7 niveles, lo único que hay que hacer es añadir el selector:
            .stretchStyleX{
            Formato_que_se _desee;
        }
*/
  color: #32658A;
}
        .stretchStyle2{
```

```
        color: #3A6E35;
}
    .stretchStyle3{
        color: #855C40;
}
    .stretchStyle4{
        color: #7E4085;
}
    .stretchStyle5{
        background-color: rgb(247,177,220);
}
    .stretchStyle6{
        background-color: rgb(247,157,220);
}
    .stretchStyle7{
        background-color: rgb(247,137,220);
}

    #metadataInfo{/*  Los metadatos, en este caso no se muestran */
        display: none;
}
```

I.5.2 Hoja de estilos del libro generado: stretch.css

El aspecto o la presentación del libro generado se puede configurar modificando esta hoja de estilos. A continuación, se va a explicar su contenido:

```
        .stretchblock:after{    /*Contenedor de los componentes de stretchText */
    padding-left: 3px;
    padding-right: 3px;
}

        .stretchblock.hidden:after{
    padding-left: 3px;
    padding-right: 3px;
}

        .stretchable { /* Parte o contenido que se muestra  u oculta del stretchText */
            text-decoration: none;
    • webkit-transition: opacity 0.2s ease-out;
    • moz-transition: opacity 0.2s ease-out;
    • o-transition: opacity 0.2s ease-out;
  transition: opacity 0.2s ease-out;
}

        .stretch { /* texto que sugiere al lector el contenido que se mostrará si se clica sobre él.
*/
    border: 1px dashed gray;
    color: inherit;
    text-decoration: none;
}

        .stretch:after{ /* Formato antes del stretch. Borde punteado e icono de color verde */
```

```
    border: 1px dashed gray;
    content: “\00bb”;
    color: green;
    padding-left: 3px;
    padding-right: 3px;
}
```

.stretch.hidden:after{ { /* Formato antes del stretch cuando se muestra. Borde punteado e icono de color rojo */

```
border: 1px dashed gray;
content: “\00ab”;
color: red;
border-bottom:1px dotted #9999CC;
text-decoration:none;
font-style:italic
}
```

div.stretchblock, div.stretch, div.stretchable{ /* Muestra los stretchblock, stretch y stretchable como un elemento inline (como un span)*/

```
    display: inline;
}
```

div.p{ /* muestra div.p como un span*/

```
    display: inline;
}
```

.stretchStyle1{ { /* A continuación se formatean los niveles o capas de profundidad de *stretchtext*. Si se añaden más de 7 niveles, lo único que hay que hacer es añadir el selector:

```
    .stretchStyleX{
    Formato_que_se _desee;
}
```

```
 */
   background-color: #EBF1F5;
}
        .stretchStyle2{
            background-color: #F4EBF5;
}
        .stretchStyle3{
            background-color: #EDEDD8;
}
        .stretchStyle4{
            background-color: #F5EFEB;
}
        .stretchStyle5{
            background-color: rgb(247,177,220);
}
        .stretchStyle6{
            background-color: rgb(247,157,220);
}
        .stretchStyle7{
            background-color: rgb(247,137,220);
}
        .p{ /* formato de los PÁRRAFOS*/
    text-indent:0.3in;
    font-family:"Times New Roman",
    Times,serif;
```

```
    font-size:100%;
    margin-top:0.01in;
    margin-bottom:0.01in;
}
        #dedication{ /* formato de la dedicatoria */
            background-color: #ffffe6;
}
        #copyright{ /* formato del copyright*/
            background-color: #ffe6ff;
}
        #prologue{  /* formato del prólogo*/
            background-color: #f2e6ff;
}
        #chapter{ /* formato de los capítulos */
            background-color: #e6f2ff;
}
        #epilogue{ /* formato del epílogo */
            background-color: #ffe6e6;
}
        #biography{ /* formato de la biografía */
            background-color: #eeffe6;
}
        #colophon{ /* formato del colofón */
            background-color: #e6e6ff;
}
```

```
        #backover{ /* formato de la cubierta*/
            background-color: #fff2e6;
}

        #metadataInfo{ /* Los metadatos en este caso se muestran ocultos */
            display: none;
}
```

I.6 Instalación

I.6.1 Instalación en Windows

Descargar el archivo comprimido que contiene todos los ficheros necesarios para la instalación en cualquier carpeta del ordenador, por ejemplo en el escritorio. Una vez descargado, descomprimir el fichero zip.

Aparecerán dos archivos ejecutables, que son:

- jdk-8u92-windows-x64.exe (para Windows de 64 bits) o jdk-8u92-windows-i586 8 (para Windows de 32 bits).
- ilsaEditor.exe

Se debe instalar en primer lugar el fichero con nombre jdk-8u92-windows-x6. La instalación será simple, ya que habrá que pulsar en el botón “next” o “siguiente” en todos los casos.

Una vez finalizado, se procederá a instalar el editor (fichero con el nombre, ilsaEditor.exe).

Para comenzar, se tiene que abrir el fichero ejecutable. En la primera pantalla aparecerá la pantalla de bienvenida al asistente de instalación (Figura I.41). Para continuar, se debe pulsar en el botón “Siguiente”.

Figura I.41 Ventana inicio de la instalación, perteneciente al proceso de instalación de ILSAditor.

A continuación, se muestra la pantalla de elección del directorio de instalación de la aplicación (I.42). Se recomienda instalarlo en la carpeta de **Documentos**. Una vez cambiado el

directorio de destino a C:\NOMBRE_USUARIO\Documentos, se pulsa en el botón de “Instalar” para proceder a la instalación.

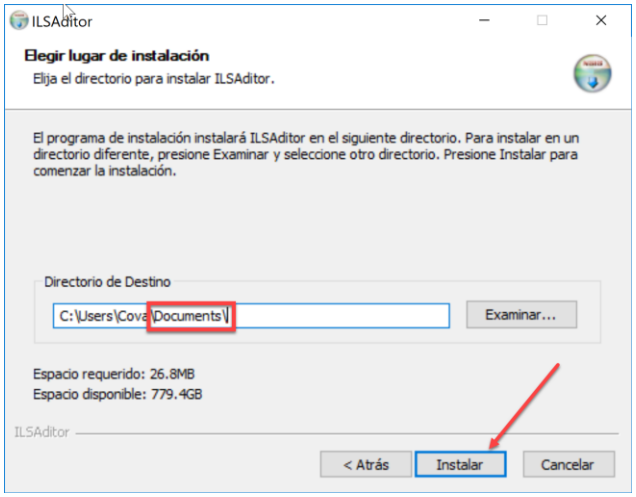


Figura I.42 Ventana de selección del directorio de instalación de ILSAditor.

Una vez finalizada la instalación, sólo queda pulsar en el botón “Terminar”, para poder cerrar el instalador y abrir el editor de forma automática (Figura I.43).

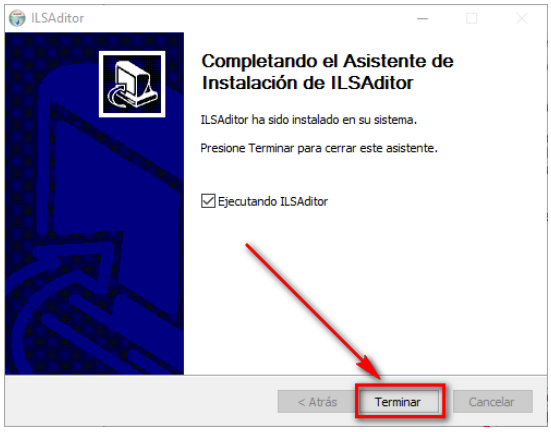


Figura I.43 Ventana de finalización de la instalación, perteneciente al proceso de instalación de ILSAditor.

I.6.2 Instalación en Linux

Descargar el fichero jar (ILSAditor.jar) y la versión Java SE Development Kit **8u92** que se ajuste a las características de hardware de tu equipo en

http://www.oracle.com/technetwork/java/javase/downloads/java-archive-javase8-2177648.html

Para instalarlo se puede seguir las instrucciones de la siguiente URL https://www.java.com/es/download/help/linux_x64_install.xml

Posteriormente, para ejecutar la aplicación se debe introducir el siguiente comando en el terminal:

/usr/java/jdk1.8.0_92/bin/java -jar ILSAditor.jar

Siendo el primer path, en donde se encuentra el java.exe de la versión 1.8.**92** que se acaba de instalar.

I.6.3 Instalación en Mac

Descargar virtual box de la siguiente URL https://www.virtualbox.org/wiki/Downloads

A continuación, añadir a virtual Box la imagen de disco virtual Ubuntu.vdi.